VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH A IMPLEMENTACE INFORMAČNÍHO SYSTÉMU PRO TVORBU E-SHOPŮ NA MODELU DROPSHIPPING

DESIGN AND IMPLEMENTATION OF INFORMATION SYSTEM FOR E-SHOP CREATION BASED ON DROPSHIPPING MODEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Pavel Hodoval**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Luhan, Ph.D., MSc**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Pavel Hodoval**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Jan Luhan, Ph.D., MSc**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh a implementace informačního systému pro tvorbu e–shopů na modelu dropshipping

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Na základě analýzy stávajícího stavu systému navrhnout řešení informačního systému s důrazem na zvýšení jeho efektivity, a to prostřednictvím změny jeho architektury a s aplikací principů projektového řízení.

Základní literární prameny:

BRUCKNER, T., J. VOŘÍŠEK, A. BUCHALCEVOVÁ a kol. Tvorba informačních systémů: Principy, metodiky, architektury. 1. vyd. Praha: Grada Publishing, 2012. 360 s. ISBN 978-80-247-4153-6.

HAYES, M. a A. YOUDERIAN. The Ultimate Guide to Dropshipping. 1st ed. Morrisville: Lulu Publishing Services, 2013. 132 p. ISBN 978-1-4834-0181-2.

KŘIVÁNEK, M. Dynamické vedení a řízení projektů: Systémovým myšlením k úspěšným projektům. 1. vyd. Praha: Grada Publishing, 2019. 208 s. ISBN 978-80-271-0408-6.

PRIES, K. and J. QUIGLEY. Scrum Project Management. 1st ed. Boca Raton: CRC Press, 2010. 198 p. ISBN 978-1-4398-2517-4.

SCHWALBE, K. Řízení projektů v IT: Kompletní průvodce. 1. vyd. Praha: Computer Press, 2011. 632 s. ISBN 978-80-251-2882-4.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. 2. aktualiz. a rozš. vyd. Praha: Computer Press, 2010. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně dne 29.2.2020

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstrakt

Diplomová práce se zabývá analýzou současného stavu informačního systému na vytváření eshopů na modelu dropshipping a efektivity vývojového týmu. Cílem práce je na základě této analýzy navrhnout a implementovat technická vylepšení a zavést vybrané principy projektového řízení.

Abstract

This thesis is aimed to analyze the current state of information system for eshop creation based on dropshipping model and the efficiency of the developement team. Goal of this thesis is to propose and implement technical improvements based on this analysis while using chosen project management principles.

Klíčová slova

Informační systém, dropshipping, eshop, databáze, projektové řízení, optimalizace, Lewinův model, Kanban

Keywords

Information system, dropshipping, eshop, database, project management, optimization, Lewin's model, Kanban

Citace

HODOVAL, Pavel. *Návrh a implementace informačního systému pro vytváření eshopů na modelu dropshipping*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská. Vedoucí práce Ing. Jan Luhan, Ph.D., MSc

Návrh a implementace informačního systému pro vytváření eshopů na modelu dropshipping

Prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 15. května 2020

.....

Pavel Hodoval

Poděkování

Rád bych poděkoval svému vedoucímu, panu Ing. Janu Luhanovi, PhD., MSc. za poskytnutí odborné pomoci při vypracování této diplomové práce.

Obsah

Úvod

Internetový prodej prostřednictvím eshopů je v dnešní době extrémně rozšířeným a populárním způsobem výdělku i nákupu, spousta podniků, které dosud prodávaly své zboží pouze v kamenných prodejnách, postupně přesouvají svoje podnikání online. Ne každý však disponuje vlastním zbožím nebo dostatečným kapitálem pro výrobu, skladování a nabídku vlastního zboží. Z toho důvodu vznikl nový fenomén v podobě dropshippingu, kde prodávající nemusí vlastnit sklady, zboží a dokonce ani vyřizovat objednávky, slouží pouze jako prostředník a vše za něj vyřizuje dodavatel.

Protože se již několik let podílím na vývoji a provozu informačního systému pro vytváření eshopů právě na modelu dropshipping, rozhodl jsem se tomuto tématu věnovat v rámci mé diplomové práce. Cílem této práce je zanalyzovat současný stav informačního systému a na základě toho navrhnout a implementovat potřebná vylepšení a opravy za pomoci vybraných principů projektového řízení.

Cíle práce, metody a postupy zpracování

Jak již bylo zmíněno v úvodu, hlavním cílem této práce je provést analýzu současného stavu informačního systému pro vytváření eshopů na modelu dropshipping Dropohs.cz a s ohledem na špatný technický stav navrhnout na základě této analýzy vylepšení, opravy a opatření pro zlepšení současné situace, které budou následně implementovány s využitím vybraných principů projektového řízení pro zvýšení efektivity vývojového týmu.

V úvodní části práce provedu studium odborné literatury z oblastí, o které se tato práce opírá. Následně představím informační systém a samotnou službu Dropohs.cz a provedu analýzu současného stavu nejen informačního systému, ale také problémů vývojového týmu vyplývajících z absence projektového řízení či nevhodně zavedených workflow. Analýzu budu stavět na ověřených metodikách a zjištěné nedostatky pokud možno kvantifikuji pro budoucí srovnání. Součástí analýzy bude také analýza rizik současného stavu.

Provedenou analýzu budu prezentovat majiteli služby a společně sestavíme požadavky na zlepšení. Od těch se bude odrážet návrhová část práce, která se zaměří na jednotlivé zjištěné problémy a jejich cílená opatření. K řízení změn bude využit Lewinův model. Tyto změny opět podrobím analýze rizik. Následně proběhne samotná implementace navržených změn a představení dosažených výsledků, kde srovnám původní stav a implementovaná zlepšení.

Kapitola 1

Teoretická východiska práce

V této kapitole se zaměřím na představení základních pojmů z oboru informačních systémů, proberu používané technologie a důležité vývojové přístupy a popíšu základní principy projektového řízení.

1.1 Informační systémy

Informačnímu systému se bude věnovat značná část této práce, proto je nezbytné si tuto problematiku blíže představit.

Informační systém (IS) lze chápat jako jakýkoliv systém uspořádaných informací. V informatice si můžeme IS definovat například jako soubor lidí, technických prostředků a metod (programů), zabezpečujících sběr, přenos, zpracování, uchování dat, za účelem prezentace informací pro potřeby uživatelů činných v systémech řízení. [14] [8] Z hlediska přístupu dělíme informační systémy na 2 základní typy:

- **Podnikové informační systémy (EIS – Enterprise Information System)** jsou přístupné pouze zaměstnancům daného podniku s různými druhy oprávnění.
- **Veřejné informační systémy** poskytují informace širší veřejnosti a o jejich provoz se často starají nejrůznější instituce (například státní orgány).

1.1.1 Podnikové informační systémy

Jedná se o systémy, které firmy využívají pro podporu a vykonávání podnikových procesů. Dělíme je na 3 základní druhy:

- **Univerzální systémy** – nejrozšířenější skupina IS. Jsou koncipovány tak, aby vyhovovaly a pokrývaly velkou škálu požadavků a potřeb, to se projevuje zejména v množství funkcí a konfigurovatelných parametrů. Podniky obvykle tyto systémy nakupují jako hotové produkty a případně s dodavatelem řeší nutná přizpůsobení. Patří sem například systémy typu ERP[1], MIS[2], CRM[3] či BI[4].
- **Systémy určené pro speciální účely** – některé podniky mají natolik specifické požadavky, že by se jim nákup a následná zásadní úprava již stávajícího univerzálního IS nevyplatila. Obvykle existuje vícero organizací, které mají stejné netradiční potřeby a proto existující specializované IS pro daný typ organizace. Jedná se například o systémy pro soudní administrativu, školy či univerzity.
- **Systémy na míru** se vyvíjí dle konkrétních požadavků zákazníka. Obvykle se jedná o nejnákladnější a nejrizikovější typ IS z důvodu běžných projektových rizik.

1.2 Přístupy vývoje IS

Pro vývoj nejen informačních systémů, ale softwaru obecně, lze využít několika modelů a metodik. Výčtem a popisem těch nejznámějších se zabývá právě tato kapitola. Vývojové modely jsou důležité z hlediska řízeného a efektivního vývoje softwaru.

[1] Enterprise Resource Planning (podnikové plánování zdrojů)
[2] Management Information System
[3] Customer Relationship Management (řízení vztahů se zákazníky)
[4] Business Intelligence

1.2.1 Vodopádový model

Vodopádový model představuje sekvenční vývojový proces, kde na sebe jednotlivé fáze přesně navazují. Rozšířil se zejména v 70.–80. letech. Proces je prováděn v následujícím pořadí:

1. Specifikace požadavků
2. Návrh
3. Implementace
4. Integrace
5. Testování (validace)
6. Údržba

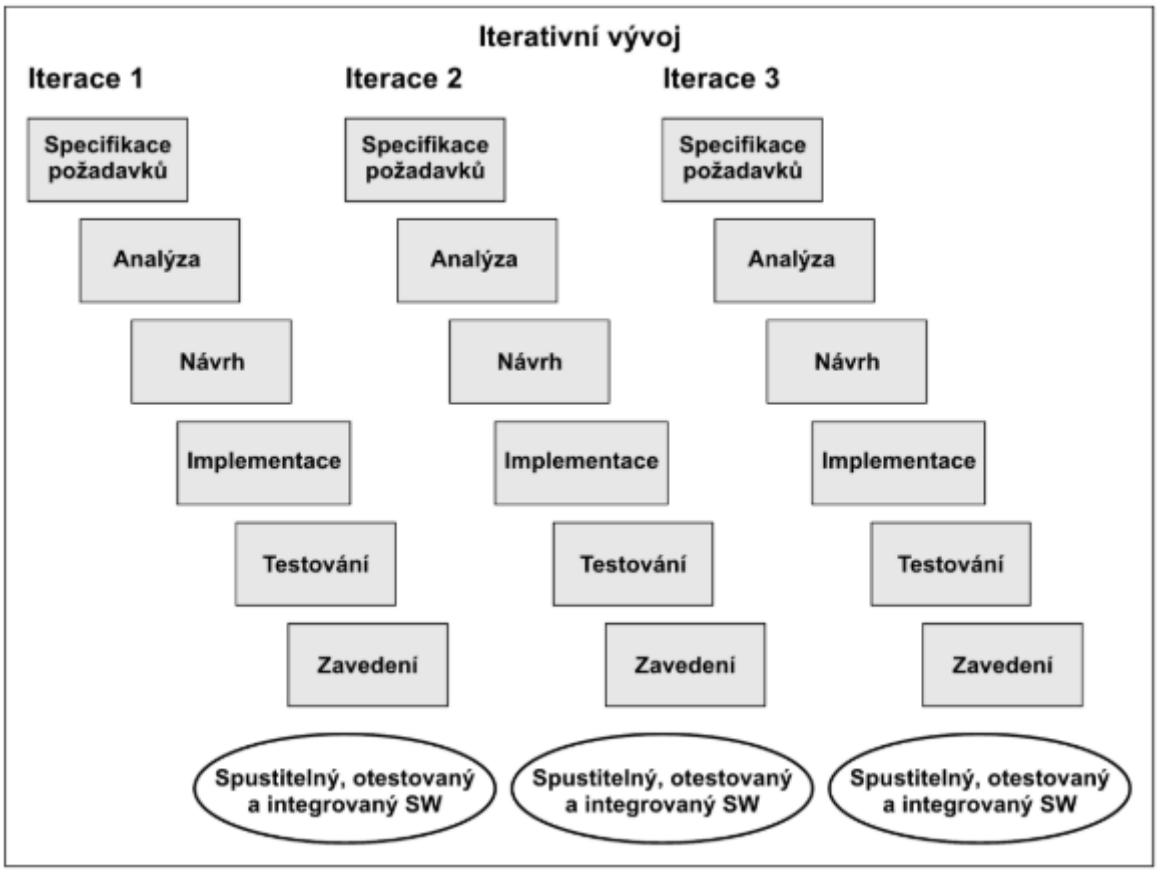
Velkou nevýhodou vodopádového modelu je, že klient často není schopen přesně stanovit všechny požadavky hned na začátku projektu, nebo své požadavky mění v průběhu vývoje. Vodopádový model vývoje v tomto ohledu postrádá flexibilitu a podobné změny požadavků můžou vést ke zpožděním a navyšování nákladů. V opačném případě, kdy je možné ve fázi specifikace požadavků stanovit naprosto všechny požadavky klienta, má vodopádový model velmi dobré výsledky. [1]

1.2.2 Iterativní model

Iterativní model vývoje je moderním přístupem k vývoji softwaru a odstraňuje některé nevýhody vodopádového modelu. Iterativní vývoj je založen na faktu, že člověk lépe řeší menší části velkého problému – iterace. Každá iterace obsahuje svým způsobem celý vodopádový proces (obrázek 1.1). Výsledkem každé iterace musí být funkční část systému, která je testována a integrována s předchozími iteracemi. [1]

Iterativní model vede ke snížení rizik při vývoji, protože již v první iteraci je možné prověřit schopnosti týmu projekt vyvíjet a správně integrovat. Dochází zde k velkému snížení rizika nespokojenosti zákazníka, protože po každé iteraci má k dispozici část softwaru, na který může podat zpětnou vazbu vývojovému týmu.

Iterativní model do značné míry využíváme pro vývoj informačního systému, kterému se tato práce věnuje.



Obrázek 1.1: Ukázka iterativního modelu, zdroj: [1]

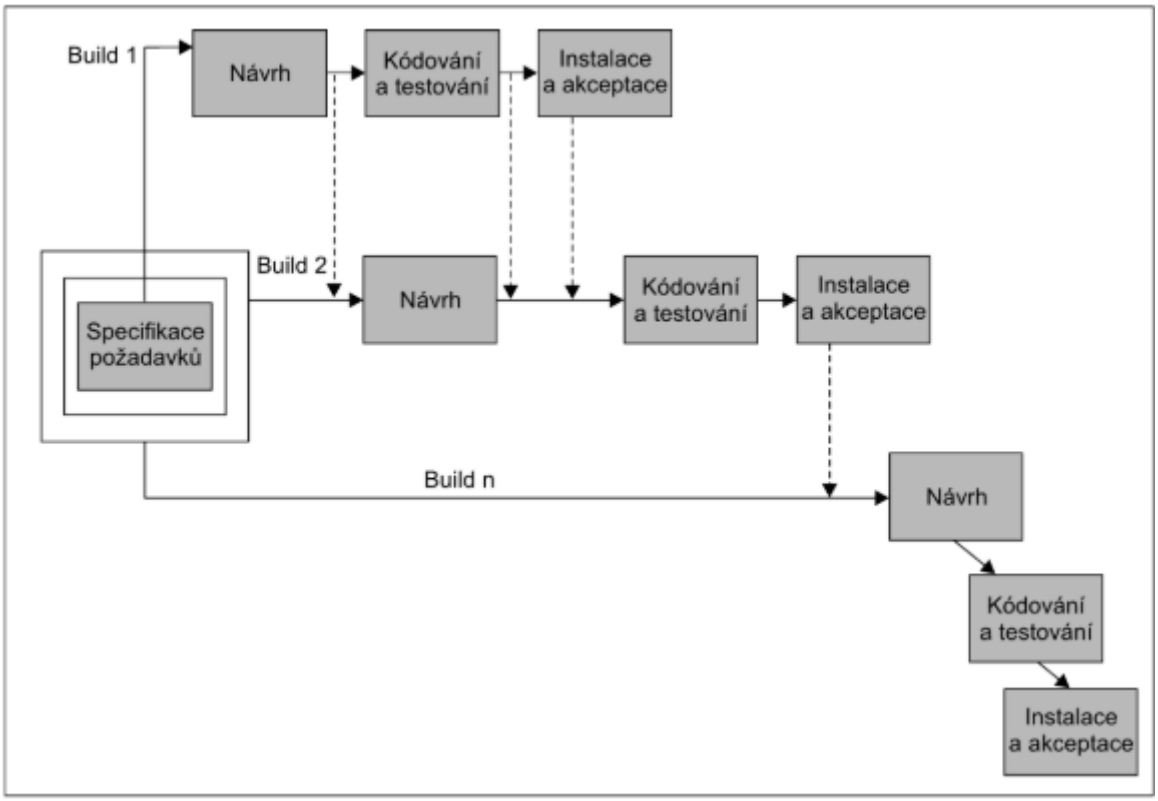
1.2.3 Inkrementální model

Inkrementální (přírůstkový) model je postaven na iterativním modelu. Definuje na začátku hrubé požadavky na systém, potom jej rozdělí na samostatné dílčí části (přírůstky) a každý přírůstek prochází všemi fázemi vývoje (vizte obr. 1.2).

Výhody inkrementálního modelu vychází z iterativního, na kterém staví – části řešení se zavádějí rovnoměrně v průběhu projektu a klient tak může sledovat, zda se vývoj ubírá správným směrem a podávat včasnou zpětnou vazbu. [1]

1.2.4 Spirálový model

Tento model taktéž vychází z vodopádového modelu, ale je postaven na opakovaných cyklech. Na rozdíl od mnoha dalších modelů tento obsahuje taktéž analýzu rizik a využívá iterativní přístup. Jeden cyklus spirálového modelu probíhá následovně:



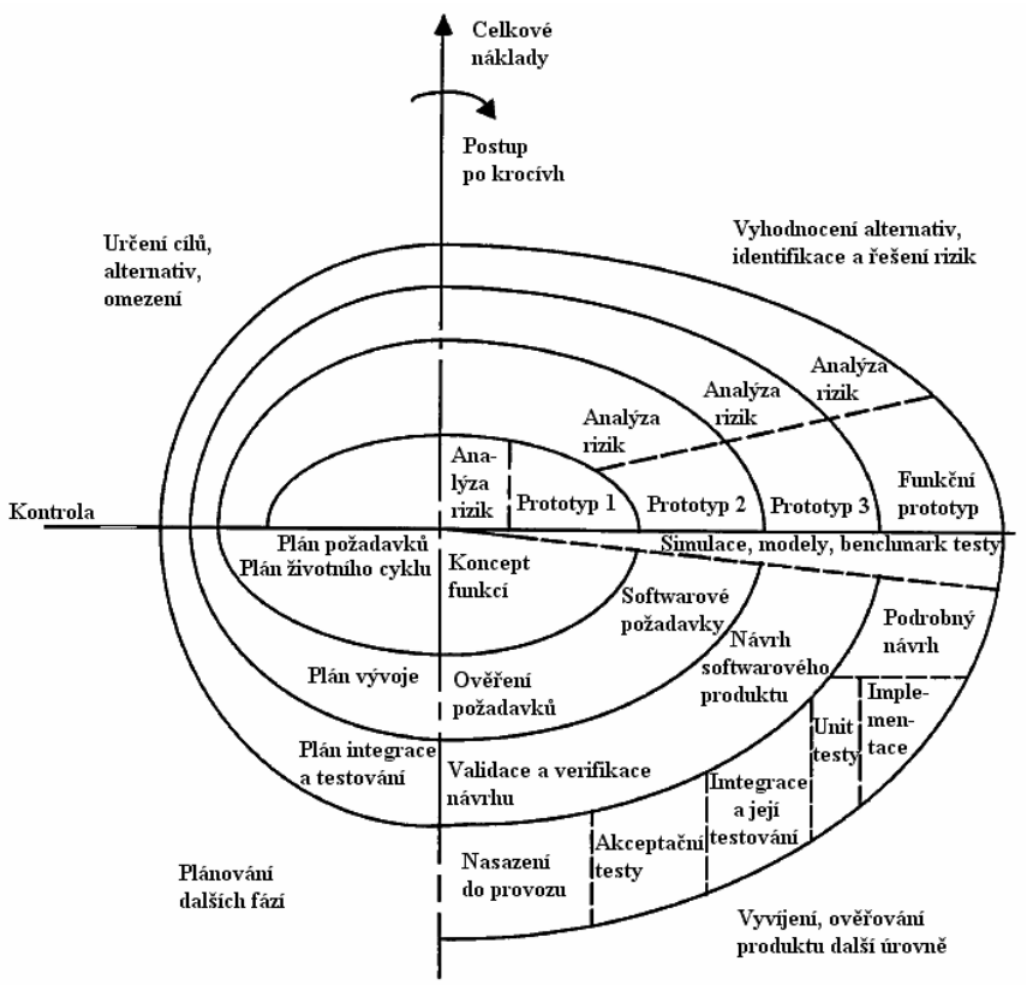
Obrázek 1.2: Ukázka inkrementálního modelu, zdroj: [1]

1. Stanovení cílů
2. Analýza rizik
3. Realizace
4. Plánování (dalšího cyklu)

Na obrázku 1.3 je k dispozici ukázka spirálového modelu.

1.2.5 Prototypový model

Je založen na principu vytváření nekompletních verzí daného softwaru (prototypů). Často se zakomponovává do ostatních vývojových modelů spíše než že by figuroval jako samostatný a úplný vývojový model. Klade si za cíl snížit klasická projektová rizika (např. změnu požadavků v průběhu vývoje) rozdělením celého procesu na menší části. Klient je zapojen do celého procesu vývoje a tím se zvyšuje šance na dodání takového softwaru, který splňuje jeho požadavky.



Obrázek 1.3: Ukázka spirálového modelu, zdroj: http://testovanisoftwaru.cz

1.2.6 RAD model

RAD (Rapid Application Developement) je moderním agilním přístupem vývoje softwaru, který se zaměřuje na přizpůsobení požadavků s ohledem na vývoj situace. Na rozdíl od vodopádového modelu, který má striktní plán, se RAD zaměřuje na rychlý vývoj pomocí podpůrného softwaru a na zpětnou vazbu [19]. Mezi výhody tohoto vývojového modelu patří především:

- Vysoká flexibilita a přizpůsobitelnost situaci, kdy vývojáři mohou při změně požadavků rychle reagovat
- Rychlé vývojové cykly snižující vývojový čas
- Podporuje znovupoužitelnost kódu, což snižuje nutnost manuálně vyvíjet nové části, to má za následek menší prostor pro vznik chyb a urychluje testování

- Zvýšení spokojenosti zákazníka (leč vyžaduje větší zákaznickou spolupráci v průběhu vývoje)
- Lepší risk management, zainteresované strany mohou diskutovat nedostatky a problémy při běhu vývoje
- Menší šance na nečekané situace díky brzkému integračnímu procesu

RAD probíhá v pěti základních fázích:

1. Definice požadavků
2. Tvorba prototypů (cyklus)
3. Sběr zpětné vazby
4. Testování
5. Prezentace výsledků

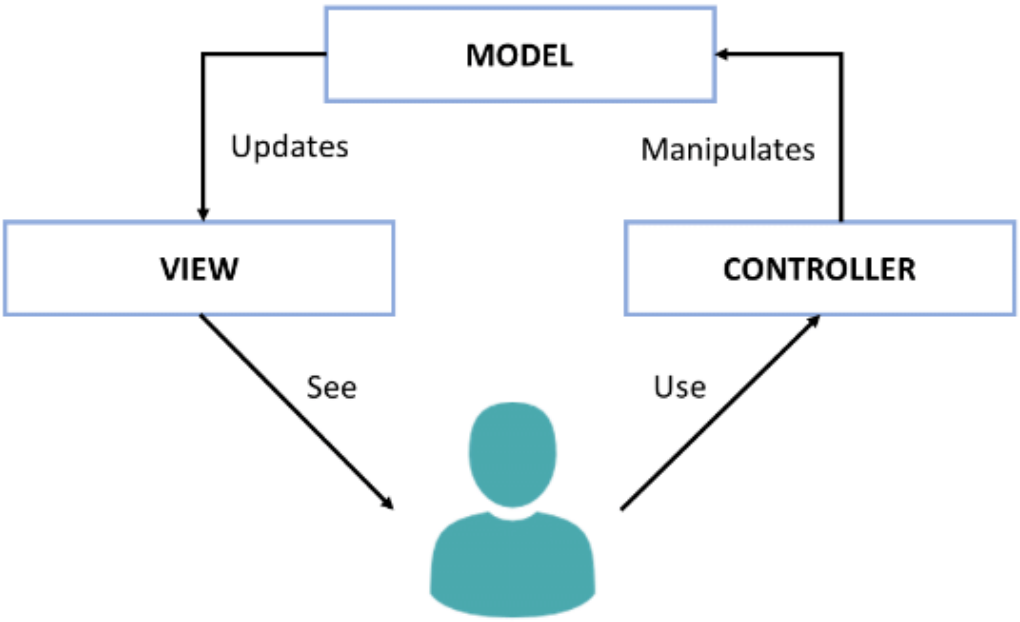
1.3 Návrhový vzor MVC

MVC (neboli Model-View-Controller) je návrhový vzor hojně využívaný u vývojových frameworků, webových aplikací a tím potažmo i informačních systémů. Princip je založen na rozdělení systému na tyto 3 části:

- **Model** – business logika a data
- **View** – pohled, například uživatelské rozhraní, výstup dat pro strojové čtení aj.
- **Controller** – tzv. řadič, který přijímá požadavky (obvykle od uživatele) a na základě nich komunikuje s modelem a případně upravuje prezentovaný pohled

Tento přístup umožňuje přehledně oddělit logiku od prezentační vrstvy a usnadňuje případné úpravy, kde postačí pozměnit pouze určitou komponentu nezávisle na těch ostatních. Základní komunikační schéma MVC znázorňuje obrázek 1.4.

Tento návrhový vzor využívá i náš informační systém, o kterém budu pojednávat v kapitole 2.



Obrázek 1.4: Ukázka komunikace MVC komponent, zdroj: http://researchgate.net

1.4 Relační databáze

Tato práce se do značné míry věnuje tématu databází, proto je nezbytné, abych provedl úvod do této problematiky, který bude sloužit jako opěrný bod pro další kapitoly.

Obecná definice databáze může být taková, že se jedná o uspořádanou soustavu dat. Podle této definice lze považovat za databázi například papírovou kartotéku. Nad kartotékou se prováděly obdobné operace jako nad dnešními elektronickými databázemi, nabízela možnosti vcelku snadného uspořádání dat a vložení nových na příslušné místo. Problémem kartoték však byly velké nároky na místo a při větších úkolech bylo nutné provést často zdlouhavou manuální práci.

Relační model databází, jehož koncept v roce 1970 navrhl E. E. Codd, se hojně využívá dodnes. Relační model je postaven na principu, kde jsou data uložena v tabulkách. Řádky tabulky představují samotné datové záznamy. Některé sloupce jsou společné pro více tabulek a vytváří vztahy (relace) mezi jednotlivými datovými tabulkami. [12]

Pro účely univerzálního dotazování v rámci relačních databází vyvinula společnost IBM jazyk SQL (Structured Query Language). Tento jazyk využívá většina relačních databázových systémů, leč syntaxe se může do jisté míry lišit pro různé systémy.

Data uložená v tabulkách v relační databázi by měla splňovat určité požadavky (tzv. normální formy) – především by měla být atomická, rozdělena do příslušných sloupců. Například sloupec `Zákazník`, ve kterém bude uloženo *Jan Novák, nar. 1993*, je příklad špatného návrhu. Správný návrh by měl mít sloupce `Jméno`, `Příjmení`, `Rok narození` a data patřičně rozdělena. Každý sloupec má předem určenou charakteristiku a datový typ – například textový řetězec (o určité délce), celé číslo, apod. V tabulce by také neměly existovat prázdné řádky.

1.5 Využívané technologie pro vývoj IS

V této podkapitole proberu několik základních technologií hojně využívaných k vývoji IS. Zmíněné technologie také využíváme v rámci našeho informačního systému, je tedy důležité se s nimi seznámit pro pochopení probírané problematiky.

1.5.1 HTML

HTML (HyperText Markup Language) je standardní značkovací jazyk využívaný pro vytváření webových stránek/aplikací a formátování jejich obsahu a určení celkové obsahové struktury. Jazyk je tvořen množinou značek (*tagy*), které jsou uvozeny ostrými závorkami < a > a určují sémantický význam obsahu, který se mezi značkami nachází. Značky se dělí na párové a nepárové, přičemž párové jsou uvozeny otevírací a uzavírací značkou v následujícím tvaru: `<značka>obsah elementu</značka>` a nepárové obsahují pouze jednu značku: `<značka>` [6] [16].

Značky také mohou obsahovat libovolný počet atributů, jenž určují další vlastnosti pro daný element, zapisují se následovně:

```
<znacka atribut1="hodnota atributu" atribut2="dalsi hodnota">
obsah elementu
</znacka>
```

Atributy mohou být různého charakteru – přes identifikátor elementu (`id`), jeho třídu (`class`) až po pomocné atributy jako `title`, který určuje vyskakovací titulek po najetí myši na daný HTML element.

Nejnovější verze standardu HTML – verze HTML5 se již dnes těší masivní podpoře v rámci desktopových i mobilních prohlížečů.

1.5.2 CSS

CSS (anglicky Cascading Style Sheets) je standardní jazyk pro popisování vizuálních vlastností (X)HTML prvků na stránce nebo XML dokumentu a používá se pro vytvoření grafické podoby stránky a rozmístění jednotlivých prvků na ní [6] [13].

Syntaxe kaskádových stylů sestává z množiny pravidel, každé pravidlo tvoří tzv. selektor a deklarační blok. Ukážeme si, jak taková syntaxe vypadá na následujícím příkladu:

```
selektor {
deklaracni blok
}
```

Selektor slouží pro určení toho, jaký element bude dané pravidlo ovlivňovat. Je možné využít více elementů naráz, čímž určíme bližší lokaci daného elementu – především jeho zanoření. Například zápis `div p` bude definovat vlastnosti pro všechny odstavce (tag `<p>`), jež jsou potomky elementu `<div>` v libovolné hloubce zanoření. Existují také speciální operátory mezi jednotlivými elementy, které tyto vztahy upřesňují, např. pokud k předešlému příkladu přidáme mezi elementy operátor `>`, tedy: `div>p`, znamená to, že se odkazujeme pouze na přímé potomky `<p>` v elementech `<div>`.

Pro jednodušší správu a rozšiřování stylopisů je možné použít například technologii SASS (Sassy CSS), která nabízí značně rozšířenou funkčnost oproti základnímu

CSS – například vkládání dalších souborů (inkluze), definování proměnných, funkce pro změnu sytosti barev, vnořené zápisy nebo tzv. mixiny, což jsou v podstatě ekvivalenty funkcí z většiny programovacích jazyků. Hlavní zdrojový soubor SASS je pak nutné přeložit do čistého CSS a výsledkem je jeden jediný CSS soubor, který lze rovnou uložit v komprimovaném tvaru.

1.5.3 JavaScript

Jedná se o interpretovaný jazyk, který umožňuje provádět na stránce interaktivní operace a změny be nutnosti znovu načíst celou stránku – dokáže v reálném čase měnit DOM strukturu stránky a nabízí tak možnosti vytvářet interaktivní webové aplikace [6] [17].

Interpret jazyka JavaScript má každý webový prohlížeč implementován vlastním způsobem, nicméně všechny dodržují zavedené W3C standardy pro společnou kompatibilitu. JavaScript se díky tomu dá často využít na ulehčení zátěže serveru tím, že některé výpočty/operace přenecháme provést klienta.

1.5.4 jQuery

[5]jQuery je javascriptový framework s otevřeným zdrojovým kódem a stal se postupem let základem téměř každé webové aplikace. Obsahuje rozsáhlou knihovnu funkcí, které umožňují psát javascriptový kód úsporněji, přehledněji a s vyšší efektivitou [2]. Navíc je možné pro jQuery velice jednoduše vytvářet zásuvné moduly a na Internetu jsou jich tisíce a většinou šířené pod open-source licencí.

Díky jQuery lze často vyčerpávající zápisy v JavaScriptu napsat velice jednoduše a elegantně. Například vestavěný systém selektorů je syntakticky inspirován tím z CSS, nejlepší bude uvést srovnání na příkladu. Toto je kód v čistém JS pro skrytí `div` elementu s id `ele`:

```
document.getElementById('ele').style.display = 'none';
```

[5] https://jquery.com/

A zde je totožná operace napsaná v jQuery:

```
$('#ele').hide();
```

Mezi další užitečné funkce patří například vestavěný systém animací a nastavování CSS vlastností a vestavěná podpora pro technologii AJAX.

AJAX (Asynchronous JavaScript ad XML) umožňuje zasílat asynchronní HTTP požadavky na libovolné soubory/skripty na stejné doméně (bezpečnostní opatření) a jQuery nabízí knihovnu pro snadnou práci s tímto mechanismem. V době moderních interaktivních webových aplikací je tedy technologie AJAX nepostradatelným nástrojem pro vývojáře.

1.5.5 PHP

[6]PHP (HyperText Preprocessor) je interpretovaný jazyk od společnosti Zend. To obecně znamená, že není nutné výsledný zdrojový kód (skripty) nijak kompilovat do binární podoby, PHP je přeloží znovu při každém spuštění. Může fungovat jak v rámci příkazové řádky na serveru, tak v kombinaci s podporovaným webovým serverem, kde webový server podle konfigurace převádí dané požadavky na PHP interpret (například Apache, Nginx) [6].

Díky jednoduchému použití a přehledné syntaxi je to jeden z nejrozšířenějších jazyků pro tvorbu backendu webových aplikací a za léta vývoje prošel mnohými změnami, kdy od verze PHP5 začal plně podporovat OOP. V současné době je dostupná již verze PHP7.x, která přináší spolu se Zend OPCache optimalizátorem drastické navýšení výkonu a rychlosti zpracování. PHP má spoustu vestavěných rozšíření, ty nejdůležitější jsou ve výchozím stavu povolené. Nabízí například rozšíření pro práci s MySQL databází, formáty JSON a XML, kryptografické funkce a spoustu dalších.

1.5.6 MariaDB

MariaDB[7] představuje open-source relační SQL databázové řešení postavené na populární technologii MySQL, za jejímž vývojem stojí někteří původní vývojáři.

[6] http://php.net/

[7] https://mariadb.org/

MariaDB nabízí všechny standardní úložné enginy jako MySQL (MYISAM, transakční InnoDB apod.) a přináší i vlastní v podobně Aria, XtraDB aj. Samotný SQL jazyk je ve většině případů plně kompatibilní s MySQL, což umožňuje snadný přechod. Výhodou MariaDB je právě otevřený zdrojový kód a poměrně rychlá inovace technologií oproti MySQL.

MariaDB Foundation, společnost stojící za tímto projektem, taktéž vyvinula technologii MariaDB Cluster, která umožňuje replikaci MariaDB databází jakožto jednotlivých uzlů a tím zajišťuje vyšší redundanci, bezpečnost a přístupnost dat. Tuto technologii taktéž využíváme v rámci našeho systému pro využití výše zmíněných benefitů.

1.5.7 Apache

Apache HTTP server je open-source webový server dostupný pro několik platforem včetně systémů založených na bázi UNIXu a systému Windows [4]. Díky své robustnosti, škálovatelnosti a rozšiřitelnosti se jedná o jeden z nejpopulárnějších webových serverů současnosti. V základu slouží k servírování statického webového obsahu, ale díky mnohým rozšířením a modulům je možné jej obohatit například o přímou podporu servírování dynamického obsahu generovaného pomocí PHP skriptů. Celý web server je plně konfigurovatelný a nabízí tak svoje funkce jak začátečníkům, tak pokročilým serverovým správcům, kteří si jej mohou přizpůsobit pro složitější serverovou infrastrukturu aj.

1.5.8 LiteMS framework

LiteMS je objektově orientovaný MVC framework napsaný převážně v jazyce PHP, který jsem sám vyvinul pro jednoduchou tvorbu nejrůznějších informačních systémů, webových aplikací a serverových procesů. Důvodem vytvoření vlastního frameworku byl především požadavek na jednoduchost, odlehčenost (od toho slovo *Lite* v názvu) a vysoký výkon. Většina moderních frameworků nabízí obrovské množství funkcí, často však za cenu přehnané komplexnosti nebo nižšího výkonu aplikace. LiteMS umožňuje díky jednoduché code base vytvářet i netradiční systémové architektury.

Framework nabízí vlastní knihovny pro efektivní práci s databází či vytváření šablon včetně funkcí jako automatické generování formulářů, tabulek, aj.

1.5.9 Git

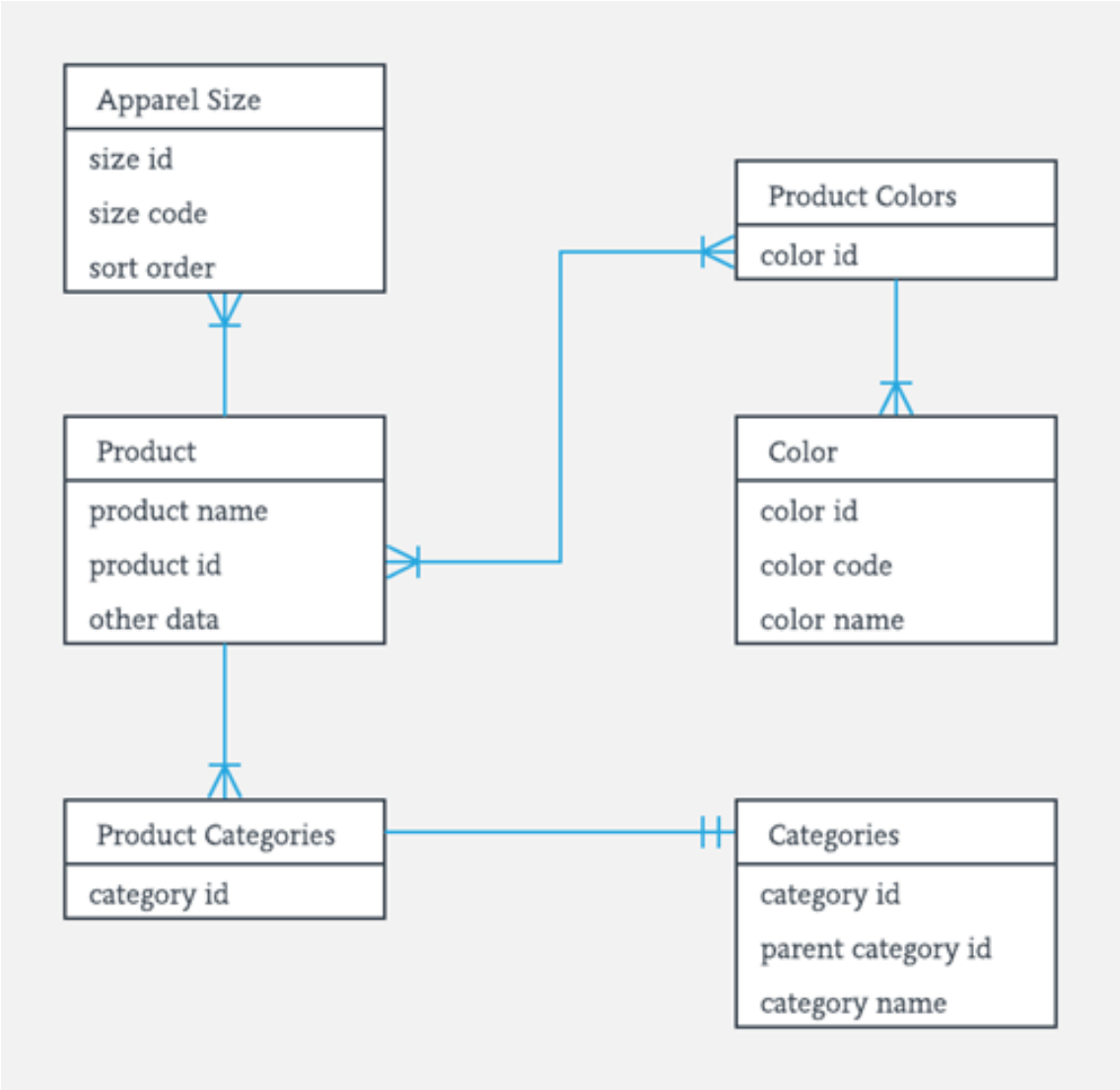
Git představuje populární open-source nástroj pro správu zdrojového kódu a jeho verzování. Jeho primárním cílem je usnadnit vývojový cyklus softwaru. Git umožňuje svým uživatelům vytvářet, využívat a přepínat mezi vývojovými větvemi (*branches*), to usnadňuje izolaci různých vývojových částí systému. Velkou výhodou Gitu je taktéž pojetí lokálních změn – k těm se chová stejně jako k těm sdíleným (dochází k verzování a jsou k dispozici všechny operace nad historií) a izoluje je do doby, než se vývojář rozhodne změny publikovat do sdíleného prostředí. [11]

1.6 Entitně relační modelování

ERM (Entity-Relationship Model) se využívá ke konceptuálnímu znázornění dat, obvykle v relační databázi. Výstupem této metody jsou tzv. ER diagramy. Ty slouží ke specifikaci požadavků na způsob a architekturu uložení dat, se kterými bude systém pracovat.

Diagram je složen z entit (běžně objekty reálného světa převedené na datovou podobu), které jsou vzájemně propojeny nejrůznějšími relacemi (vztahy). Každá relace navíc na obou jejich koncích definuje tzv. kardinalitu, která určuje typ relace. Nejznámějšími typy kardinality jsou 1:1, 1:N a M:N (a všechny opačné varianty) – například kardinalita 1:N značí, že jednomu záznamu z tabulky A může náležet neomezeně záznamů z tabulky B (1 uživatel může mít N objednávek). K ukázce jednoduchého ER diagramu slouží obrázek 1.5.

ER diagramy budou využity v rámci této práce pro znázornění databázové architektury představeného informačního systému.



Obrázek 1.5: Ukázka ER diagramu, zdroj: http://guru99.com/

1.7 Průběžná integrace

Průběžná integrace (anglicky Continuous Integration – často se používá zkratka CI) je technika vývoje softwaru založená na principu, kdy každý člen vývojového týmu často *integruje* svoji práci (například nahraje své změny do verzovacího systému) – většinou minimálně jednou denně.

Integrace je následně ověřena automatizovaným systémem, který provede sestavení softwaru (tzv. build) a následně jej otestuje (např. pomocí jednotkových testů) na přítomnost integračních a dalších chyb. Tento způsob vývoje umožňuje týmům rychle odhalovat vzniklé problémy a nedostatky a urychluje tak celkový vývoj. [3]

Díky rozvoji cloudových služeb se průběžná integrace stala velmi oblíbeným vývojovým aspektem a v dnešní době nabízí širokou škálu funkcí, kromě testování je například možné provádět kontrolu kvality kódu apod.

1.8 Projektové řízení

Jelikož je cílem práce aplikovat vybrané principy projektového řízení pro budoucí vývoj, je důležité, abychom si představili základy projektového řízení a některé konkrétní metody.

Projektové řízení je proces, ve kterém organizace efektivně využívají své omezené zdroje k realizaci projektů. Za provádění projektu a jeho výsledky je zodpovědný projektový manažer a tým, který vede, řídí, ošetřuje případná rizika a využívá příležitosti. Projektové řízení je flexibilní a kvalifikovaná reakce na nejrůznější události, které v průběhu času v projektu nastávají. [7] Dle [18] lze projektové řízení definovat jako aplikaci znalostí, dovedností, nástrojů a technik na projektové aktivity s cílem dosáhnout požadavků projektu. Zainteresované strany jsou lidé zapojení či dotčení projektovými aktivitami.

Dle přístupu metodologie k projektovému řízení můžeme tento proces rozdělit na 2 základní typy – tradiční a agilní.

1.8.1 Tradiční přístup

Tradiční přístup k projektovému řízení je postaven na principu, kdy máme pevně definované procesy a instrukce, co v jaké situaci dělat a jakým způsobem. Na tradičním přístupu staví například metodika PRINCE2[8], která je zaměřena spíše na manažery. Naproti tomu PMBOK navíc implementuje prvky lídra, kterému dává k dispozici další nástroje. Metodika dle IPMA pro změnu ke klasickému procesnímu pohledu na projektové řízení klade také velký důraz na tzv. *soft skills* – jemné dovednosti, které se zaměřují na lidské chování.

[8] PRoject IN Controlled Environment

1.8.2 Agilní přístup

Agilní projektové metodiky se soustředí spíše na týmové řízení, spolupráci a vedení projektů, dávají tak manažerům větší volnost při práci s týmem. Asi nejznámějším zástupcem agilního přístupu je metodika Scrum.

Metodika scrum se zaměřuje především na byznys požadavky daného projektu. Odstraňuje aktivity, které nemají žádnou, nebo minimální, přidanou hodnotu. Důležitou součástí metodiky scrum je také vzdělávání se samotného vývojového týmu v průběhu projektu. Tým se na začátku každého dne schází na tzv. *sprint*, který trvá cca 10 minut – zde každý člen týmu sdílí, co dělal včera, co plánuje dělat dnes a co ho blokuje. Výhodou scrumu je flexibilní, iterativní přístup k vývoji, který umožňuje rychle a efektivně reagovat na požadavky zákazníka. Ten je v rámci této metodiky více zapojen do celého procesu vývoje. [15]

1.8.3 Metoda Kanban

Slovo kanban pochází z japonštiny a v překladu znamená *informační tabule* – jedná se o jednoduchou projektovou metodu, která staví na vizualizaci prováděné práce a rozděluje ji do menších úkolů. Každá tabule by měla obsahovat minimálně sloupce TO DO (dosud nezapočaté úkoly), WIP (work in progress – právě zpracovávané úkoly) a Done (dokončené úkoly).

Každý úkol je reprezentován kartičkou umístěnou do příslušného sloupce. Pointa metody spočívá v omezení počtu karet (úkolů) ve sloupci WIP – dokud je počet roven stanovenému limitu, nesmí být zahájen žádný nový úkol.

Dle [9] nejsou výhody metody kanban vzhledem k její jednoduchosti vidět na první pohled, avšak tento systematický a vizuální přístup organizace přináší zlepšení celkové firemní kultury a efektivity řízení projektů.

1.8.4 Lewinův model řízení změny

Provádění rozsáhlých změn v podniku je důležité nějakým způsobem analyzovat a následně řídit. Vzhledem k rozsahu změn, které budou navrženy v této diplomové

práci, je vhodné si představit metodiku, pomocí níž tyto změny budeme aplikovat řízeně a bezpečně.

„Lewinův model řízení změny je rozdělen do tří základních fází – a to na fázi **rozmrazení**, **vlastní provedení změny** a závěrečnou fázi **zamrazení**. Vlastní projekt řízení změny je obvykle modelován síťovými grafy (např. CPM, PERT, aj.).“ [10]

Před samotným zahájením procesu změny by měly být zodpovězeny následující otázky:

- Jak vypadá požadovaný stav, kterého chceme dosáhnout?
- Jaké jsou faktory změny a jak jsou intenzivní?
- Kdo bude změnu podporovat a kdo bojkotovat?
- Kde bude provedena intervence?
- Jak tuto intervenci provedeme?
- Jak celý proces dopadl?

Fáze rozmrazení slouží k přípravě změny, kdy jsou *rozmrazena* stávající pravidla a zvyklosti ve firmě. Je zde nutné provést všechny analýzy potřebné k provedení změn.

Ve fázi samotné změny proběhne naplánovaná změna. Vycházíme z provedených analýz a provádíme změny na základě doporučení, která z nich vyplývají. V rámci změny může dojít i k dočasnému zhoršení situace.

V poslední fázi dochází k fixaci provedených organizačních a dalších změn firmy či jiné organizace.

1.8.5 Skórovací metoda analýzy rizik

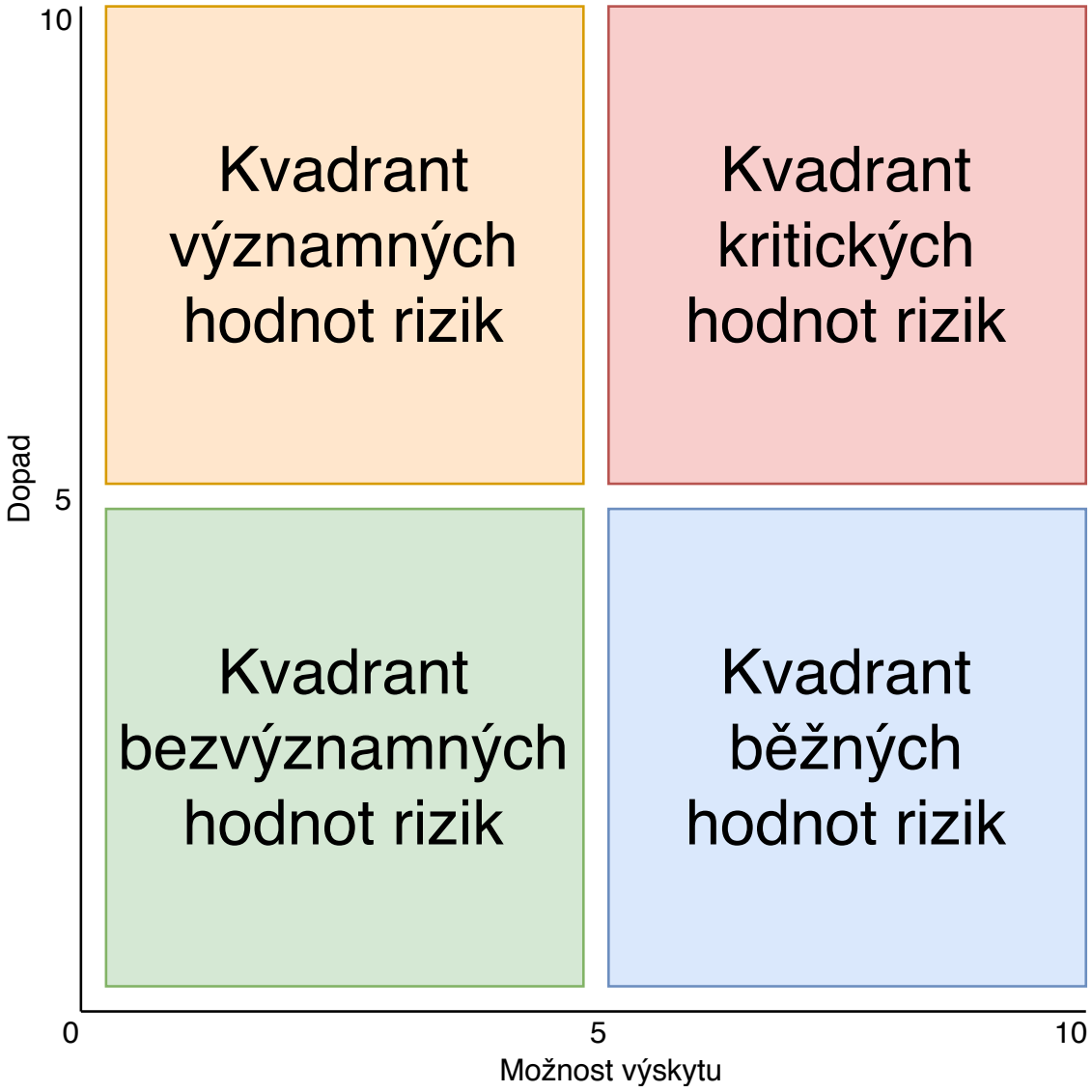
Analýza rizik jde ruku v ruce s procesem řízené změny a je jednou z nejdůležitějších disciplín projektového řízení.

Skórovací metoda analýzy rizik se skládá ze tří fází:

1. Identifikace rizika
2. Ohodnocení rizika – ohodnotíme možnost výskytu rizika a dopad rizika na škále 1–10. Hodnota rizika je součinem těchto dvou hodnot.

3. Navrhneme opatření

Na základě výše zmíněného je pak možné sestavit tzv. mapu rizik (obrázek 1.6), kde vodorovná osa představuje možnost výskytu a svislá osa dopad. Mapu rozdělíme na 4 kvadranty podle významnosti daných rizik. Obzvláště zvýšenou pozornost je nutné věnovat rizikům z kritického kvadrantu a navrhnout pro ně vhodná opatření.



Obrázek 1.6: Ukázka mapy rizik

1.8.6 Síťová analýza

Síťová analýza je dalším důležitým procesem v rámci řízení projektů s ohledem na plánování jednotlivých projektových činností a celkového harmonogramu.

Síťová analýza obecně představuje označení metod pro modelování určitého souboru činností, které je nutno provést k dosažení stanoveného cíle. Cílem síťové analýzy je uspořádat tyto činnosti tak, aby bylo dosaženo optimálního času – tedy aby byl projekt splněn v požadovaném čase.

Vizuálním výstupem síťové analýzy je síťový graf, který znázorňuje návaznost jednotlivých činností. Skládá se z uzlů a hran, které mají dle zvolené metody různý význam. V grafu následně můžeme hledat tzv. kritickou cestu, která udává nejdelší dobu projektu. Na kritické cestě leží (kritické) činnosti, které mají nulovou celkovou rezervu a jejich zpoždění by vedlo ke zpoždění celého projektu. K síťové analýze se nejčastěji používají metody CPM a PERT, které popíši v následujících podkapitolách.

Metoda CPM

CPM (critical path method – metoda kritické cesty) se používá především v případech, kdy předem známe délku jednotlivých činností. To může platit například u opakujících se úkonů, které již dříve byly provedeny a mají deterministickou délku průběhu. Tato metoda se hodí například pro stavební projekty.

Metoda PERT

Metoda PERT (PRoject Estimated Time) má uplatnění v případech, kdy neznáme přesnou délku jednotlivých činností. Jedná se tedy o stochastickou metodu, která pro výpočet předpokládané délky činnosti používá tři veličiny: optimistický odhad ($a_{i,j}$), realistický odhad ($m_{i,j}$) a pesimistický odhad ($b_{i,j}$). Samotný vzorec pro odhad délky činnosti je následující:

$$t_{i,j} = \frac{a_{i,j} + 4m_{i,j} + b_{i,j}}{6} \tag{1.1}$$

Tato metoda může být vhodná například pro IT projekty nebo výzkum, jejímu použití se budu dále věnovat v návrhové části této práce.

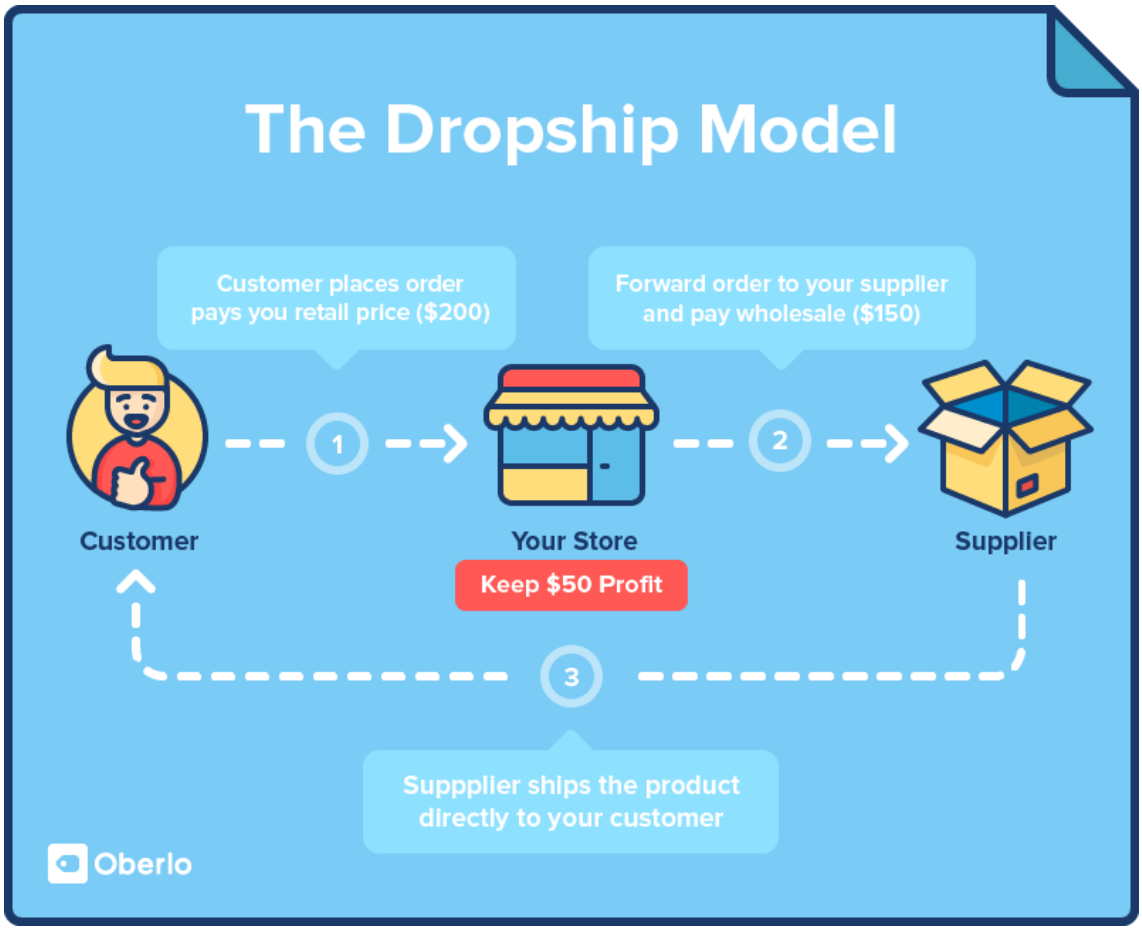
1.9 Dropshipping

Tato práce pojednává z velké části o informačním systému, který nabízí snadné vytváření a provoz eshopů na prodejním modelu dropshipping, představíme si tedy, jak tento model funguje.

Dropshipping představuje unikátní způsob maloobchodního prodeje, při němž prodávané zboží nakupuje, skladuje a expeduje zákazníkům dodavatel namísto prodejce (obrázek 1.7). Výdělkem prodejce se stávají marže za prodané zboží. Dodavatelé nasazují různé cenové restrikce na zboží, aby je prodejce neprodával za příliš nízkou či předraženou cenu.

Výhodou dropshippingu oproti běžnému obchodnímu modelu je několikanásobně nižší vstupní kapitál na rozběhnutí podnikání – lze začít jen s pár tisíci korun. Dropshipping může být využit jako přivýdělek nebo – podle investovaného času a úsilí – jako hlavní zdroj příjmů. [5]

Dropshipping nepředstavuje způsob, jak rychle a snadno zbohatnout, ale je tak často prezentován a rozmohla se díky tomu velká vlna pochybných dodavatelů a služeb. Je důležité dávat si pozor při výběru dodavatelů a služeb a také spočítat si potencionální výdělky dle marží daných dodavatelů



Obrázek 1.7: Znázornění modelu dropshipping, zdroj: https://justcreative.com/

Kapitola 2

Analýza současného stavu

V této kapitole představím službu Dropohs, její informační systém a stav projektového řízení vývojového týmu.

2.1 O službě Dropohs

Služba Dropohs[1] slouží k rychlému a jednoduchému vytváření eshopů na modelu dropshipping. Zákazníci tak mohou za měsíční paušál provozovat eshopy bez nutnosti vlastního zboží, skladu či přepravních služeb. Služba je přímo propojena se systémem Dropshippng.cz, který poskytuje různé dodavatele zboží a stará se o samotné zprostředkování objednávek a vyplácení provizí zákazníkům. Služba Dropohs je vlastněna panem Davidem Marákem, který ji provozuje jako fyzická osoba.

Služba byla poprvé spuštěna v roce 2018 a za rok a půl běhu vygenerovala svým zákazníkům více než 25 000 objednávek. Svůj eshop si zde (minimálně na zkoušku) založilo cca 1 500 zákazníků. Na službě se v současné době podílí celkem 5 osob – majitel služby, 3 vývojáři (jeden na vedoucí pozici) a jeden serverový specialista.

Zákazníci mohou eshop provozovat na vlastní doméně, taktéž si mohou zřídit podle svého tarifu emailové schránky, či za příplatek pořídit například zavedení SSL certifikátu pro zabezpečení svého eshopu. Vzhled každého internetového obchodu je možné přizpůsobit potřebám zákazníka. Více se konkrétními funkcemi systému bude zabývat kapitola 2.2.

[1] https://dropohs.cz/

2.2 Současný stav IS

Následující kapitola se věnuje popisu funkcionality celého informačního systému a jeho jednotlivých částí, technickému představení systémové architektury včetně rozvržení databáze a problémům současného řešení.

2.2.1 Představení IS

Celý informační systém je složen ze čtyř hlavních částí - veřejné prezentace celé služby, administrace celého systému (*manažerská administrace*), veřejné části konkrétních eshopů a administrace konkrétních eshopů (*shopadmin*).

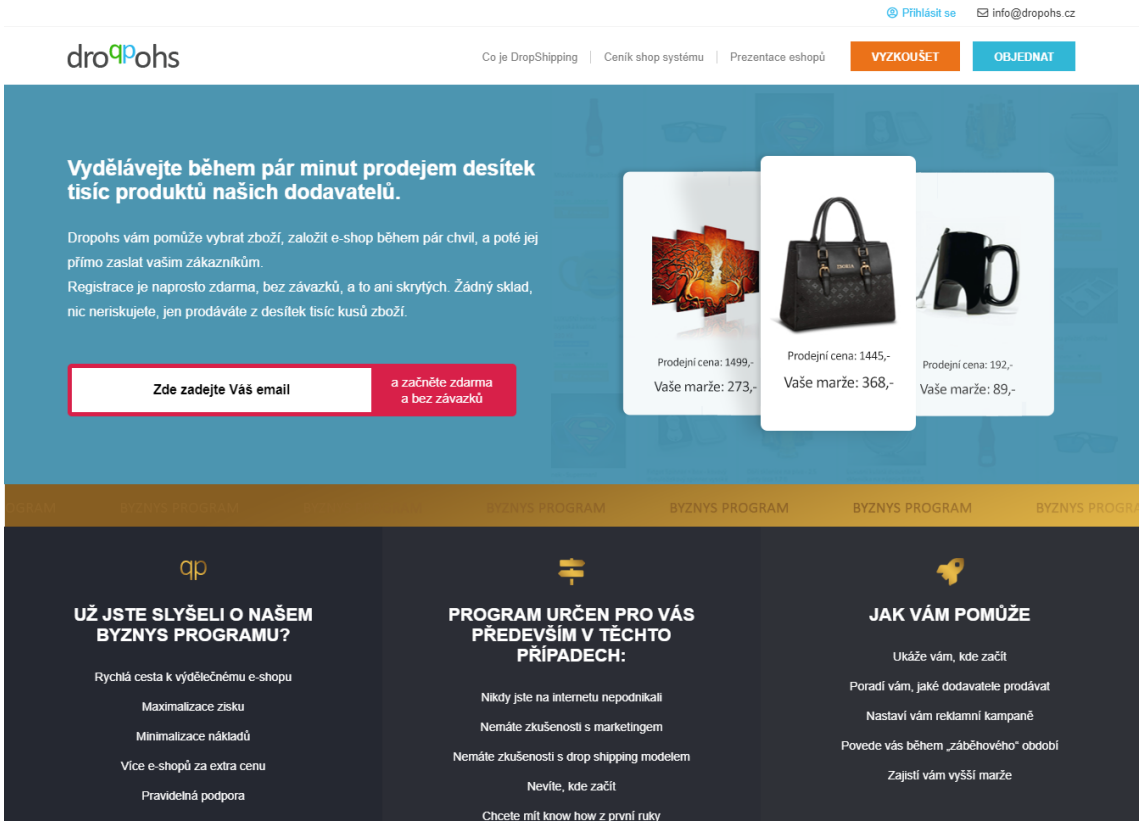
Veřejná prezentace služby

Tato část systému (dostupná na adrese `https://dropohs.cz/`) slouží především k představení samotné služby, jejích funkcí a ceníku jednotlivých tarifů. V této části je taktéž možné zaregistrovat si zákaznický účet a vytvořit si eshop – je zde možnost vyzkoušení zdarma po dobu 30 dnů. Všechny tarify jsou koncipovány formou měsíčního paušálu. Podle výše tohoto paušálu se zákazníkům zvyšují limity některých funkcí nebo přibývají kompletně nové funkce, které v nižších tarifech nejsou dostupné. K ukázce hlavní stránky slouží obrázek 2.1.

Administrace celého systému

Interně nazývaná jako *manažerská administrace*, slouží ke správě chodu celého systému. Umožňuje spravovat a sledovat stav všech eshopů a jejich majitelů, provedených plateb, vygenerovaných faktur apod. Dále nabízí možnost spravovat tarify a vytvářet nové, nastavovat jejich ceny, funkce a stanovovat, které tarify budou součástí aktivní nabídky.

Administrace dále umožňuje komunikovat se zákazníky pomocí vlastního systému zpráv pro vyřizování požadavků. K dispozici jsou také nejrůznější nastavení globálního charakteru, možnost editace informací na veřejné části webu aj.



Obrázek 2.1: Ukázka domovské stránky služby Dropohs

K dispozici je také kompletní správa všech uživatelů služby, filtrované přehledy jejich eshopů, platebních transakcí aj. Uživatelé jsou rozděleni do 4 základních skupin oprávnění:

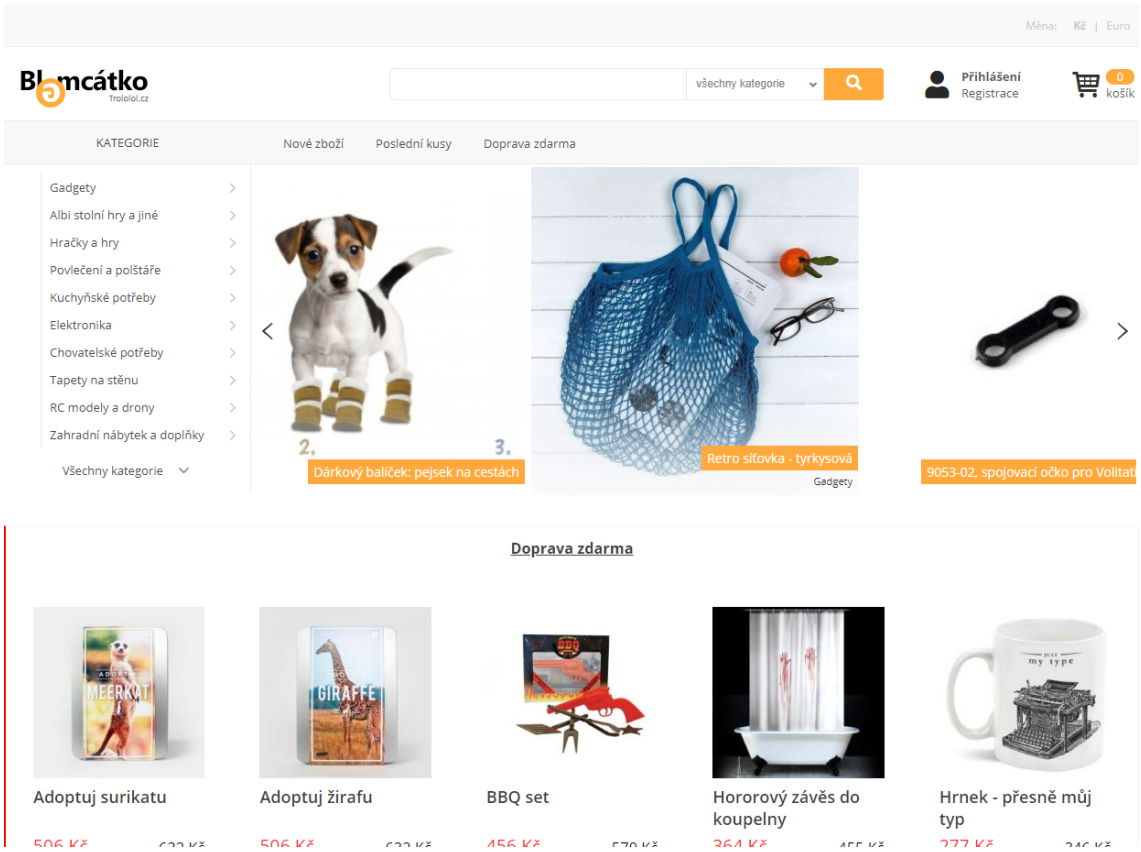
- **uživatel** – výchozí oprávnění každého nového uživatele, nemá žádný přístup do manažerské administrace
- **investor** – speciální typ oprávnění, který umožňuje osobám, jež investovaly do vzniku služby přístup k omezeným přehledům systému zobrazujícím výnosy služby a jejich aktuální podíl ze zisku
- **administrátor** – má přístup k většině částí administrace, nemůže však měnit oprávnění uživatelů
- **manažer** – nejvyšší stupeň oprávnění, který má neomezená práva

Přehledová stránka administrace pak zobrazuje aktuální stav celého systému – počet aktivních eshopů, tržby služby za daný den a měsíc, vygenerované objednávky všech zákazníků, vypovězené eshopy a předpokládaný měsíční zisk.

Veřejná část konkrétních eshopů

Veřejná část eshopu slouží k prezentaci a prodeji produktů, které se náš zákazník rozhodl prodávat. Hlavní stránka nabízí tzv. boxy produktů, které si zákazník může přizpůsobit v administraci eshopu (možné je měnit jejich pořadí, zobrazení i kritéria výběru) – jsou zde boxy z konkrétních kategorií, nebo boxy systémové (například *Novinky*, *Doprava zdarma*, atd., vizte obrázek 2.2).

Dále je k dispozici výpis všech kategorií eshopů (které je buď možné vytvářet ručně)nebo importovat spolu s produkty od dodavatele, výpisová sekce dle konkrétní kategorie, detail produktů (který lze rozšířit pomocí příplatkových modulů) a vyhledávání v produktech. Systém nabízí standardní nákupní košík, možnost registrace zákazníků a systém objednávek. Na webu se dále nachází standardní sekce jako kontakty, obchodní podmínky atp.

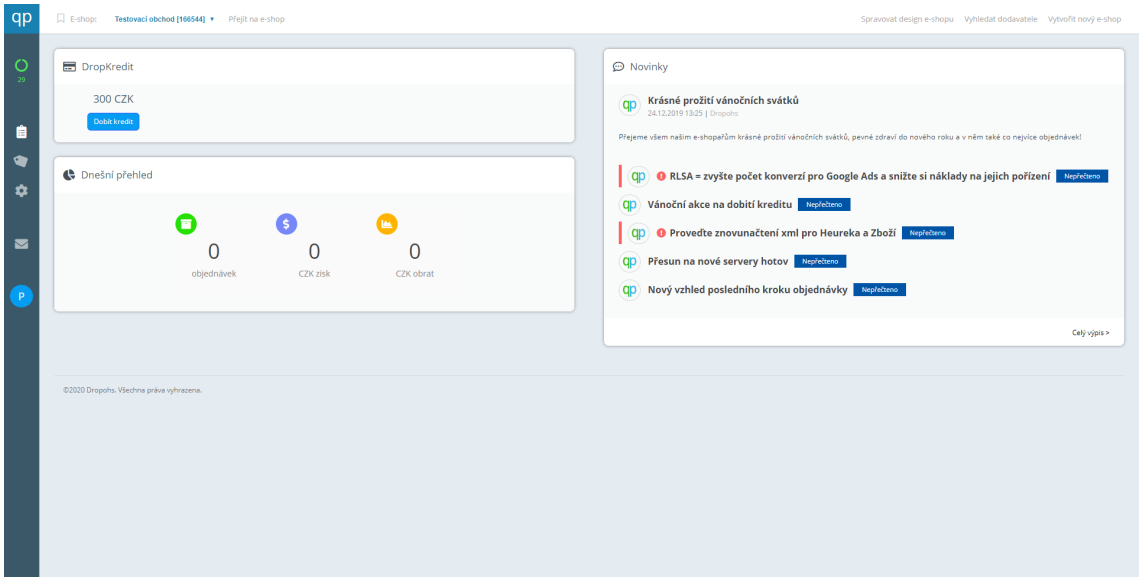


Obrázek 2.2: Ukázka hlavní stránky vygenerovaného eshopu

Administrace konkrétních eshopů

Každý registrovaný uživatel vlastnící alespoň 1 eshop má oprávnění pro přístup k administraci svých eshopů (tzv. *shopadmin*). Systém umožňuje snadnou správu všech eshopů daného uživatele a mezi jeho eshopy může zákazník snadno přepínat pomocí roletky viditelné ve všech částech systému.

Hlavní stránka eshopové administrace (obrázek 2.3) shrnuje stav tzv. DropKreditu (který slouží k nákupu doplňkových služeb), ukazuje přehled o aktuálních objednávkách, obratu a zisku. V pravé části se pak nachází sekce novinek, které zde publikuje majitel služby. V určitých případech se zde také mohou zobrazit boxy se speciálními nabídkami pro konkrétní zákazníky.



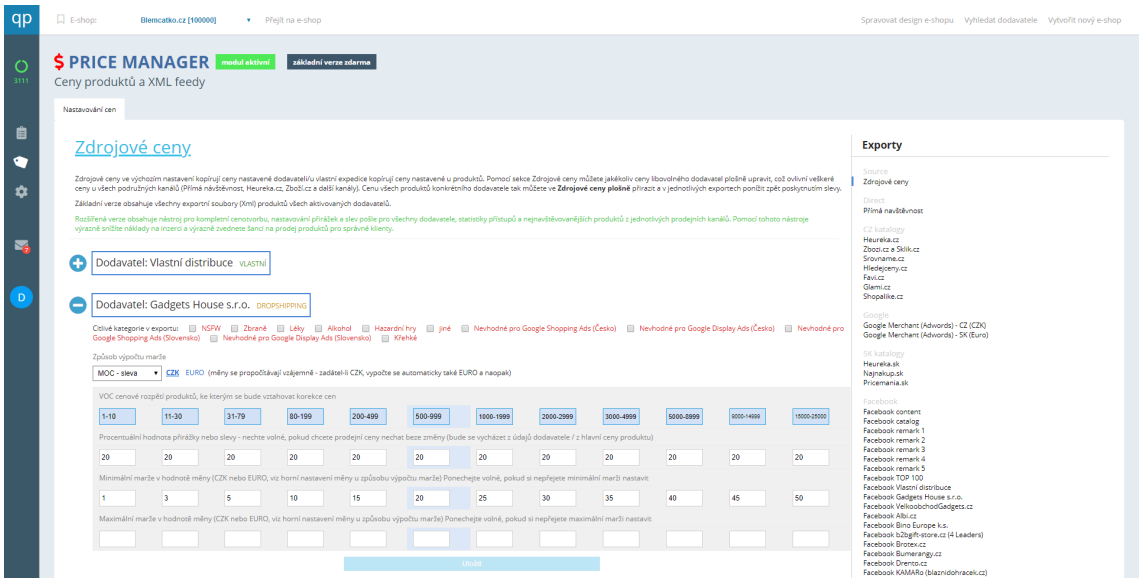
Obrázek 2.3: Ukázka hlavní stránky administrace eshopu

Administrace dále nabízí kompletní správu objednávek, včetně zobrazení jejich aktuálního stavu a informací ze služby Dropshipping.cz, která objednávky vyřizuje. Nechybí ani správa zákazníků, kteří se na daném eshopu zaregistrovali, včetně funkcí pro splnění normy GDPR.

V další části administrace může zákazník manipulovat s produkty, které na svém eshopu nabízí – editovat jejich data, nastavovat ceny (případně slevy), přidávat vlastní produkty na prodej (tzv. vlastní distribuce) či importovat/odebrat produkty

od dropshipping dodavatelů. K tomu náleží taktéž kompletní správa kategorií produktů, typy dopravy a způsoby plateb za objednávky.

Velkou částí produktové správy je taktéž tzv. *Price Manager* (obrázek 2.4) – příplatkový modul, který hromadně nastavuje marže za prodej produktů na základě velké škály pravidel. Modul funguje na principu provize z každé úspěšné objednávky. Tento modul také nabízí možnost generování XML zdrojů dat pro nejrůznější cenové srovnávače typu Heureka.cz, Google Merchant, aj. včetně nastavení specifických cen a marží pro návštěvníky přicházející z konkrétního srovnávače.



Obrázek 2.4: Ukázka správy cen a marží

Následující část administrace představuje správcovskou sekci rozdělenou na několik podsekcí:

- **Nastavení tarifu** – přehled o parametrech stávajícího tarifu daného eshopu a možnost jeho snížení či navýšení dle potřeb zákazníka
- **Správa příplatkových modulů** – výčet všech dostupných modelů s možností jejich aktivace, resp. deaktivace
- **Propojení s Dropshipping.cz** – nastavení napojení uživatele na API služby Dropshipping.cz, toto je nezbytný krok pro prodej produktů od dodavatelů
- **Fakturační údaje** – specifické pro prodej zboží v konkrétním eshopu
- **Doména a základní údaje** – zde probíhá propojení domény zákazníka se službou Dropohs, nachází se zde instrukce pro korektní nastavení DNS zá-

znamů domény a editace základních údajů eshopu (jako je název, popis, klíčová slova, kontaktní email, aj.)

- **Vzhled a šablona** – konfigurace vzhledu eshopu pomocí tzv. *rainbow editoru*, uspořádání prvků na hlavní stránce a volba jejich obsahu, nastavení sociálních sítí atd.
- **Management správců** – možnost pozvání nových, správa stávajících a detailní editace oprávnění jednotlivých správcovských skupin
- **Nastavení emailových schránek** – v současnosti funguje poloautomatizovaně na základě žádosti klienta o zřízení schránky
- **Správa emailových šablon** – editace přednastavených vzorů pro emaily přidružené k objednávkám zboží typu vlastní distribuce (zboží přímo prodávané zákazníkem)

Následující sekce administrace slouží k editaci nejrůznějších textů a sekcí eshopu – obchodních podmínek, informací o práci s cookies, reklamačního řádu, ale také vkládání kódů pro měření návštěvnosti konverzí (Google Analytics, Sklik a další). V neposlední řadě je k dispozici sekce *Zprávy*, která umožňuje zákazníkům komunikovat s technickou podporou služby a vznášet nejrůznější požadavky. K tomu je přidružená také stránka na zasílání nápadů a zpětné vazby. Poslední sekce v administraci slouží k nastavení údajů týkajících se uživatelského účtu – kontaktní a fakturační údaje apod.

2.2.2 Dropshipping model

Pro správné pochopení celé problematiky našeho IS je nezbytné si představit, jak systém funguje v rámci dropshippingového prodeje.

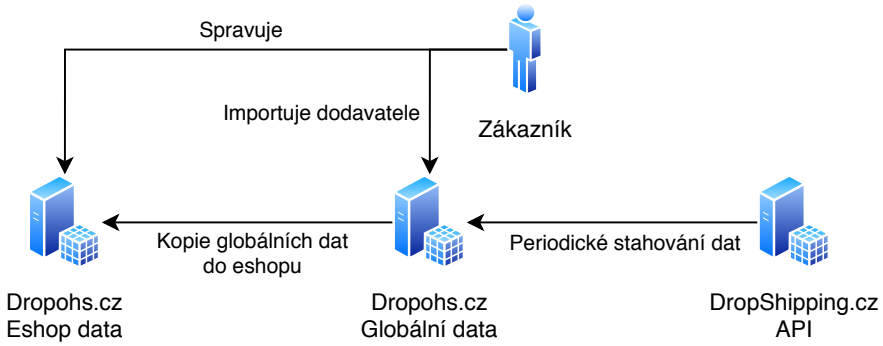
Naše služba má uzavřené partnerství s portálem DropShipping.cz, který poskytuje třetím stranám prodej zboží od smluvených dodavatelů prostřednictvím jejich API[2]. Pomocí toho API je možné získat veškerá data o produktech, obrázcích a kategoriích jednotlivých dodavatelů. Dále máme přístup také k internímu API, díky

[2]Application Programming Interface – rozhraní pro programování aplikací

kterému se uživatel přes naši službu může zároveň zaregistrovat i na DropShipping.cz a rovnou si i propojit oba účty, což je nezbytný krok pro importování dodavatelů.

Jelikož je API komunikace a transformace dat z důvodu velkého objemu časově náročná, stahujeme průběžně data od DropShipping.cz pro všechny dostupné dodavatele a ukládáme do naší globální databáze v potřebném formátu, zejména pro urychlení procesu importu (zboží) a lepší odezvu. Ze stejných důvodů k nám také ukládáme všechny externí kategorie a stahujeme veškeré obrázky. Celý tento proces je plně automatizovaný a běží periodicky jednou za den.

Zmíněný import dodavatelů představuje proces, kde si zákazník přes administraci eshopu vybere, jaké produkty a od koho by rád prodával. Je k dispozici několik kritérií, podle kterých je možné importované produkty filtrovat. Volitelně si zákazník také může importovat dodavatelem nastavené kategorie produktů, případně si je může roztřídit sám podle potřeby. Po potvrzení importu backendový proces nakopíruje veškerá zvolená data do tzv. lokálních dat eshopů. Pro lepší pochopení slouží obrázek 2.5. Tento proces více technicky popíši v kapitole 2.2.4.



Obrázek 2.5: Proces importu zboží

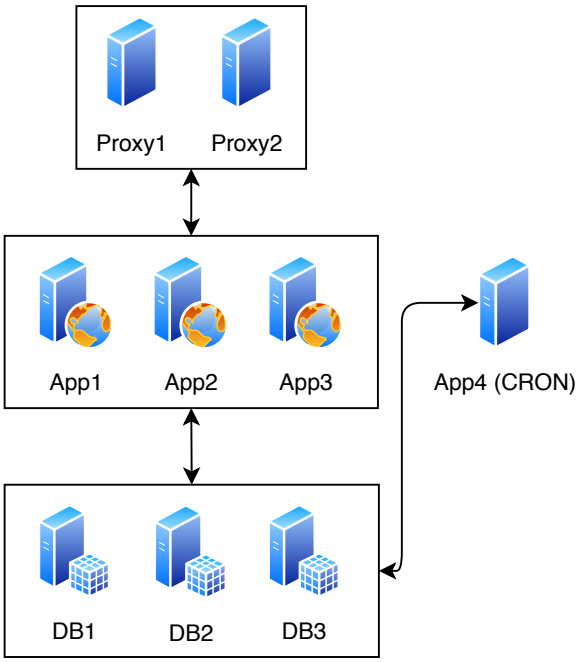
2.2.3 Serverová architektura

Celá serverová architektura je postavena na operačním systému Debian Linux. Pro servírování obsahu slouží 3 webové servery využívající technologii Apache a jeden aplikační server, který slouží ke zpracování opakujících se operací na pozadí (tzv. *CRON server*). Dynamický obsah se servírován prostřednictvím skriptovacího jazyka PHP a databáze MariaDB.

Před těmito webovými servery stojí 2 proxy servery – slouží jako vyrovnávače zátěže a delegují požadavky na zmíněné 3 webové servery dle vytíženosti každého z nich. Vyrovnávače využívají technologii HAproxy pro předávání svých požadavků.

2.2.4 Databázová architektura

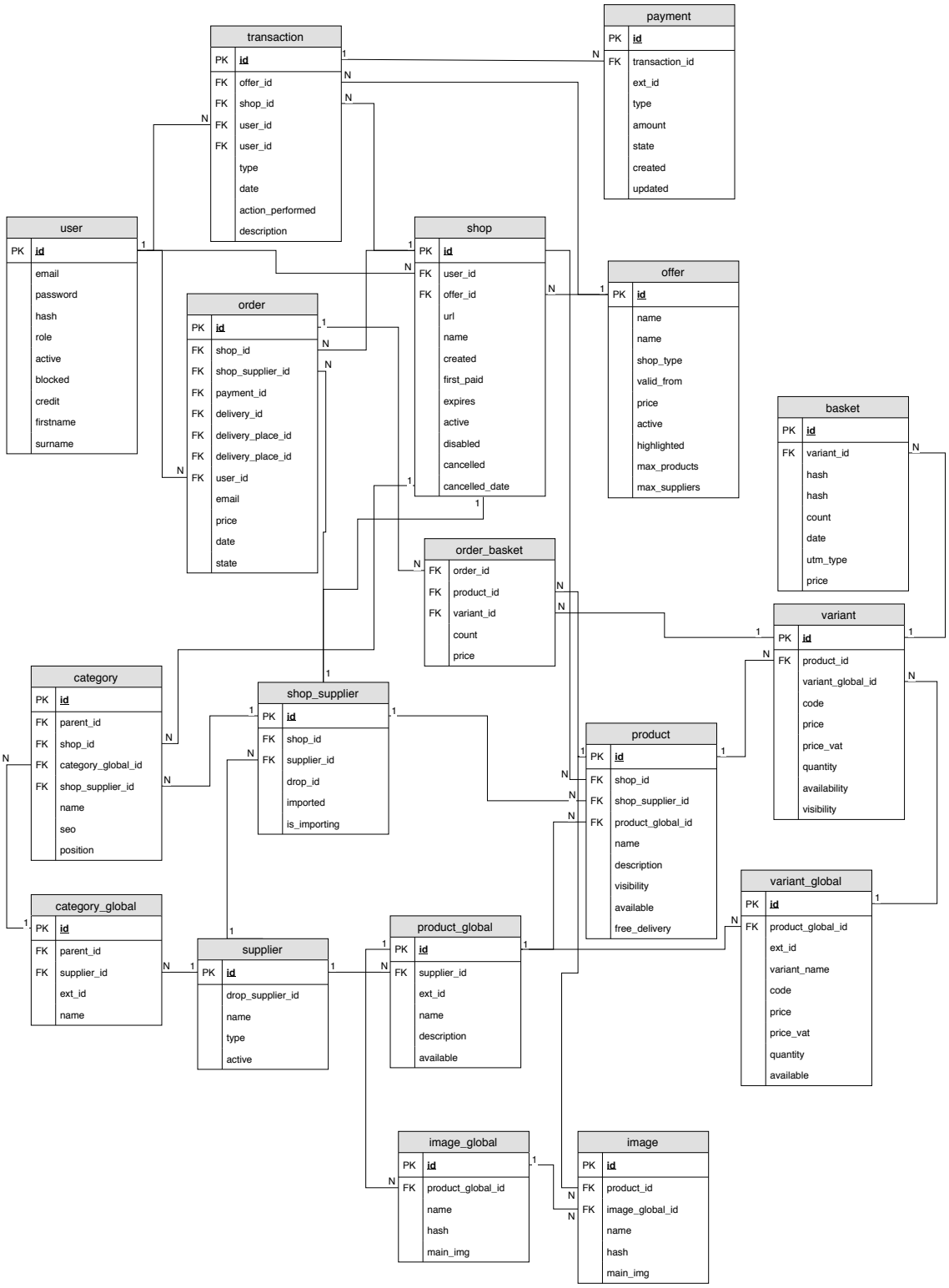
Pro přístup k databázi slouží celkem 3 databázové uzly na technologii MariaDB, které jsou replikovány pomocí nástroje MariaDB Galera Cluster. To zajišťuje redundanci, rozložení zátěže a zvýšenou dostupnost dat. Každý ze tří webových serverů primárně využívá právě jeden databázový uzel. V případě výpadku některého z uzlů webový server automaticky zvolí jiný dostupný uzel. Cluster pro svou funkcionalitu vyžaduje transakční databázový engine InnoDB, proto musí všechny tabulky být postaveny právě na něm. Schéma celé serverové infrastruktury zachycuje obrázek 2.6.



Obrázek 2.6: Serverová infrastruktura

Databáze v současné době sestává ze 108 tabulek o celkové velikosti přibližně 38 GB. Všechny tabulky jsou uloženy v jedné společné databázi s názvem `dropohs`. S ohledem na obrovské množství tabulek a sloupců u každé z nich zde představím pouze

zjednodušené schéma databáze prostřednictvím ER diagramu (obrázek 2.7). U tabulek jsem záměrně nechal pouze sloupce, které jsou pro pochopení schématu nezbytné.



Obrázek 2.7: ER diagram důležitých tabulek před změnou systému

Z diagramu je patrná sdílená povaha databáze pro všechny eshopy – většina tzv. *lokálních* tabulek obsahuje sloupec `shop_id`, který funguje jako cizí klíč a specifikuje, ke kterému eshopu daný záznam patří. To platí především pro dodavatele eshopu (`shop_supplier`), produkty (`product`), kategorie (`category`), obrázky (`image`), nastavení marží (`margin`), aj. V lokálních tabulkách tím pádem dochází k duplicitě záznamů odlišených především ID daného eshopu.

Tabulky s přídomkem `_global` pak slouží jako celková nabídka pro všechny zákazníky – například `product_global` obsahuje kompletní nabídku unikátních produktů všech dostupných dodavatelů, z této tabulky se následně kopírují záznamy do tabulek lokálních. Patří sem například i tabulky `delivery` (způsoby dopravy), `supplier` (konkrétní dodavatelé) aj.

Proces vytvoření lokálních dat funguje následovně:

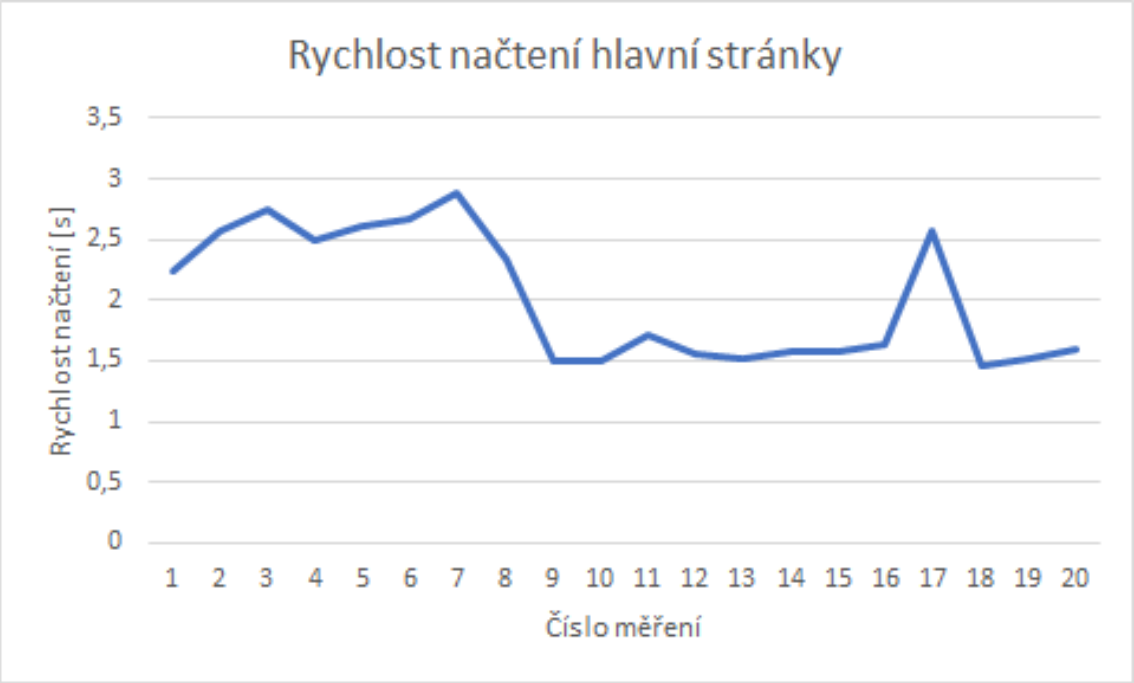
1. Zákazník si vytvoří nový eshop – vznikne nový záznam v tabulce `shop`.
2. Zákazník si naimportuje nového dodavatele – nejdříve se vytvoří záznam v tabulce `shop_supplier`, která představuje lokální kopii dodavatele včetně dodatečných nastavení (které produkty importovat podle různých pravidel, apod.).
3. CRON operace běžící cyklicky na backendu serveru dodavateli nastaví příznak `importing` a začne postupně kopírovat produkty, obrázky a kategorie z tabulek globálních do tabulek lokálních
4. Po dokončení operace se dodavateli nastaví příznak `imported`, který značí, že dodavatel obsahuje všechna potřebná data.
5. Zákazník od této chvíle může zboží prodávat.

2.2.5 Odezva systému

Pro budoucí účely srovnání dopadu provedených změn jsem provedl měření odezvy při stávajícím stavu systému. Toto testování sestává ze dvou částí – první část sleduje rychlost načítání veřejných stránek eshopu a druhá část měří dobu generování XML *feedů* pro cenové srovnavače (jako je Heureka.cz, Google Merchant, aj.).

Rychlost načítání eshopu

K tomuto testu byla použita vývojářská konzole prohlížeče Google Chrome – konkrétně karta Network – zde byla měřena doba načtení samotné stránky bez obrázků, stylů apod., které by ovlivňovaly negativním způsobem výsledek měření. Hodnoty byly testovány na referenčním eshopu Blemcatko.cz, který provozuje přímo majitel služby. V době měření čítal eshop 120 771 produktů. Jednotlivá měření byla provedena téměř bezprostředně po sobě s vynuceným obnovením cache. Zjištěné hodnoty shrnují grafy 2.8, 2.9 a 2.10.

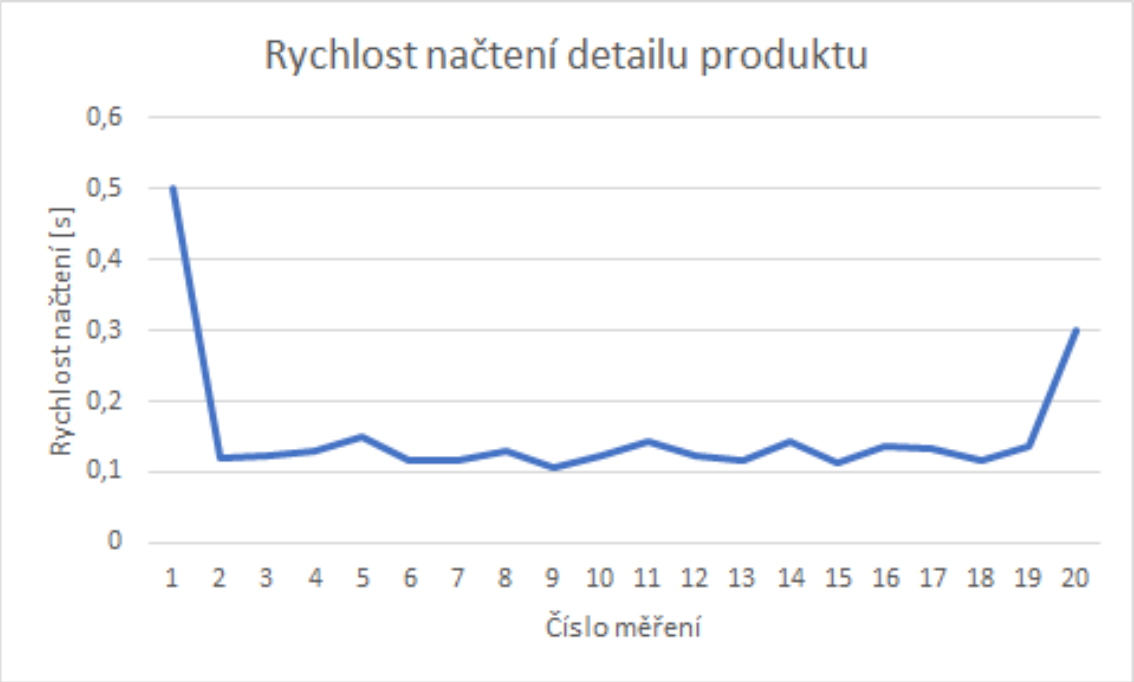


Obrázek 2.8: Rychlost načtení hlavní stránky eshopu

Rychlost generování XML feedů

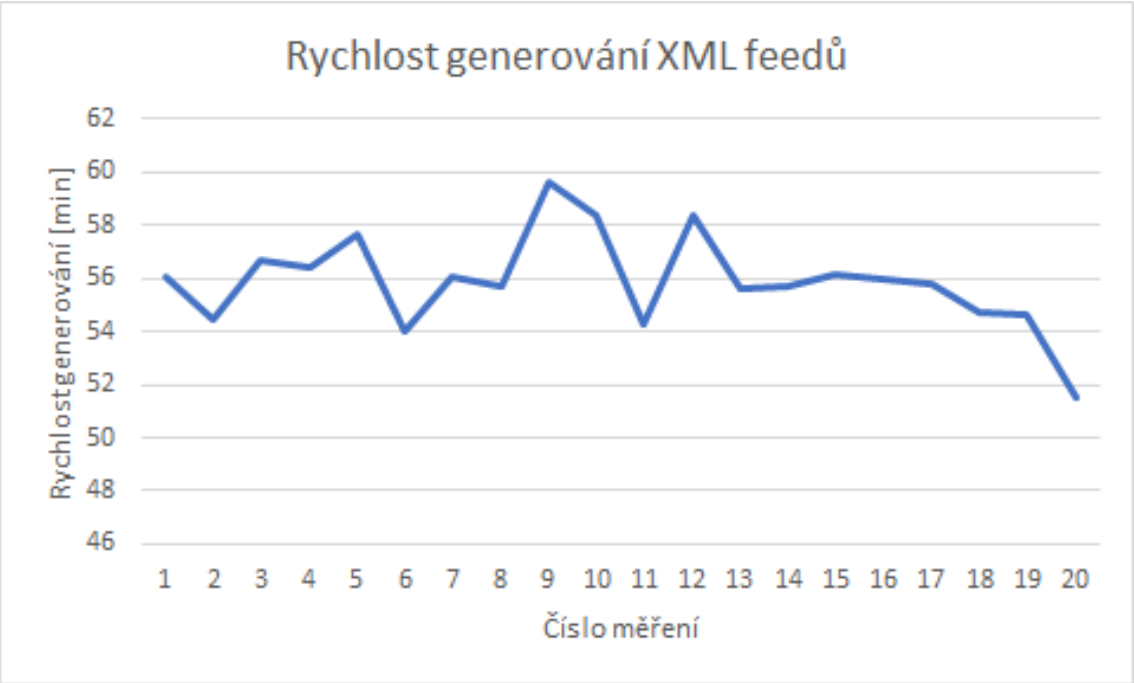
XML feedy pro cenové srovnavače se generují 2× denně pro všechny eshopy pomocí CRON operace běžící na backendu serveru. Je to časově náročná úloha, ale ve výsledku šetří výkon serverů při samotném stahování těchto předgenerovaných XML souborů srovnávači – servíruje je jako statický obsah. Měření byla založena na výstupu skriptu, který uvádí dobu generování všech feedů (cca 20 XML souborů)

Obrázek 2.9: Rychlost načtení kategorie s 2 000 produkty



Obrázek 2.10: Rychlost načtení detailu produktu

zmíněného referenčního eshopu Blemcatko.cz. Získané hodnoty jsou shrnuty grafem 2.11.



Obrázek 2.11: Rychlost generování XML feedů

2.2.6 Stav strojových testů

Ke kontrole funkčnosti a zachování konzistence systému využíváme jednotkové (unit) testy k testování jednotlivých tříd a metod. Používáme testovací framework PHPUnit, který nám umožňuje snadno spravovat a spouštět strojové testování.

Jednou z funkcí knihovny PHPUnit je také zjištění pokrytí testů (code coverage), které udává, kolik procent zdrojového kódu je ve skutečnosti testováno. Provedl jsem analýzu pokrytí testů a výsledky prezentuji v tabulce 2.1.

Metrika	Pokrytí	Poznámka
Třídy	2,13 %	Udává, kolik tříd jazyka je kompletně pokryto testy
Metody	9,47 %	Udává, kolik metod jazyka je kompletně pokryto testy
Řádky kódu	19,22 %	Udává, kolik řádků kódu je kompletně pokryto testy

Tabulka 2.1: Pokrytí jednotkovými testy

Z tabulky je patrné, že strojové testy jsou v tristním stavu a je velká pravděpodobnost vzniku plynoucích komplikací.

2.3 Stav vývojového workflow

Pro souběžný vývoj a verzování informačního systému využíváme nástroj Git na platformě GitHub.com. Zde máme vytvořený privátní repozitář, ke kterému má přístup pouze vývojový tým.

Co se týče samotného Git workflow, v současné době máme jednu hlavní větev `master`, která reprezentuje ostrou verzi systému. Nové větve (*branches*) vytváříme pouze v případě vývoje větších nových součástí nebo zásadních změn systému. Nevyužíváme mechanismus *pull requests*, všichni uživatelé mají právo provádět `merge` (sloučení) a `commit` (přidání změn) operace na hlavní větvi. Každý uživatel pak má svou vlastní `local` větev, která slouží pro usnadnění práce při konfliktech operace `merge`.

Uvedený postup rozhodně není optimální – je zde velké riziko způsobení chyb a komplikací jednotlivými uživateli, kteří mají možnost nahrávat úpravy přímo na ostrou větev. Taktéž zde chybí jakákoliv automatizovaná forma kontroly kvality.

2.4 Stav projektového řízení

V současné době se v týmu nepoužívají žádné oficiální metody projektového řízení. Úkoly se majitelem projektu zadávají hromadným emailem vývojovému týmu, v horším případě na Facebook Messenger jednotlivým vývojářům. Z mé strany bylo navrženo využívat službu Trello pro lepší management úkolů a přiřazování povinností. Tento nápad se uchytil jen částečně a stále se využívají ostatní metody zadávání práce.

Využívání komunikačních kanálů jako je Messenger způsobuje nevědomost ostatních členů týmu o aktivitě spolupracovníků, pokud si tuto informaci implicitně nevyžádají. Rozesílání úkolů emailem je poněkud zastaralé e neefektivní pro management a sledování stavu daných úkolů.

Často také dochází k impulsivní změně priorit, některé úkoly se ze dne na den zanechávají a přechází se na nové. To kumuluje nedokončené úkoly jednotlivým vývojářům a snižuje efektivnost jejich práce – příprava na práci na novém úkolu

často vyžaduje nezanedbatelný čas – například technickou konzultaci s vedoucím vývojářů a přesnější specifikaci požadavků.

2.5 Problémy současného řešení

S ohledem na představený IS, stav vývojového workflow a projektového řízení nyní shrnu problémy plynoucí ze současného stavu, rozdělené do příslušných kategorií.

2.5.1 Problémy IS

V začátcích projektu nepředstavovalo sdílení databáze pro všechna data žádný zásadní problém, protože tabulky obsahovaly málo záznamů a reakční doba celého databázového systému byla v podstatě instantní. Toto řešení bylo v dané situaci nejlevnější a nejrychlejší na provedení.

S narůstajícím počtem zákazníků, eshopů a tím pádem i produktů, jejich kategorií, obrázků atp., se **odezva databáze značně zpomalila**. To odhalilo mnohé implementační nedostatky a chybějící optimalizace jak v samotné implementaci IS, tak v návrhu relační databáze. Načítání jednotlivých sekcí eshopů se z několika desítek milisekund **protáhlo dokonce na několik vteřin**. Při velké zátěži systému pak dochází až k přečerpání časového limitu na požadavky a stránky eshopu se vůbec nenačtou. To způsobí ochromení celého systému a dočasnou nedostupnost.

Na základě vzniklých komplikací bylo provedeno mnoho optimalizací jak na aplikační úrovni, tak v rámci relační databáze. Bylo vytvořeno mnoho tabulkových indexů pro zrychlení reakční dob systému. Toto řešení sice dočasně situaci zlepšilo z pohledu rychlosti načítání, nicméně s sebou neslo další potíže vyplývající ze **špatného návrhu databáze** – začaly se objevovat tzv. *deadlocky*[3] způsobené velkým množstvím indexů nad databázovými tabulkami. Ze stejného důvodu se také značně zpomalily operace editace a mazání (`UPDATE`, `DELETE`). To způsobilo problémy například při odstranění dodavatelů z eshopů našich zákazníků – samotné odebrání dodavatele a všech jeho produktů, kategorií a obrázků nyní trvalo **až desítky minut**, přičemž byl systém extrémně zpomalený a místy dokonce nedostupný.

[3]Stav, při kterém se systém snaží zároveň číst i zapisovat a není schopen tento konflikt vyřešit

Například lokální tabulka produktů všech eshopů (`product`) v současné době obsahuje přes **2,8 milionu záznamů a 47 indexů**. Složitější dotazy nad touto tabulkou jsou velmi pomalé a neefektivní. Tabulka obrázků (`image`) pak čítá přes 10 milionů záznamů. Toto je jádro problému celého systému – s rostoucím počtem klientů a eshopů roste i velikost těchto tabulek do neúměrných rozměrů, kdy se jakákoliv práce s nimi stává neefektivní. Také dochází k velkému množství přístupů k daným tabulkám (z důvodu sdílené povahy), což způsobuje dříve zmíněné deadlocky.

2.5.2 Problémy strojových testů

Dalším nedostatkem stávajícího řešení je malé pokrytí jednotkovými (unit) testy. To znamená, že malé procento zdrojového kódu je systematicky testováno vytvořenými strojovými testy.

Vzniklo již několik situací, kdy přestaly fungovat některé funkce systému, protože po jeho změně nebylo možné otestovat konzistentní funkcionalitu, ruční testování takto rozsáhlého systému je v podstatě nemožné jak z časových, tak i finančních důvodů.

Malé pokrytí jednotkovými testy vysoce **zvyšuje šanci na vznik nových chyb, opakování se již opravených chyb či dokonce vznik nových chyb v důsledku opravování chyb jiných**. Tento stav skrývá obrovské riziko až katastrofálních potencionálních problémů, které se bez automatických testů těžko předvídají.

2.5.3 Problémy současného vývojového workflow

Zásadním nedostatkem současného vývojového procesu je absence vytyčených pravidel pro práci s Gitem. Zejména pak konvence vytváření a pojmenovávání větví, nastavení oprávnění pro hlavní větev a absence průběžné integrace.

2.5.4 Problémy absence projektového řízení

V důsledku nízké (či dokonce nulové) aplikace principů projektového řízení je práce týmu často velmi neefektivní, špatně informována vůči ostatním kolegům a obecně nevhodně koordinována a organizována.

To má za následek **zbytečné prodlužování prováděných úkolů**, konflikty mezi vývojáři a jejich úkoly a v konečném důsledku to vede ke zvyšování nákladů a snižování efektivity celého týmu.

2.6 Analýza rizik

S ohledem na problémy popsané v kapitolách výše se nyní zaměřím na analýzu rizik plynoucích z ponechání současného stavu. K analýze využiji skórovací metodu zmíněnou v kapitole 1.8.5. Tabulka 2.2 shrnuje jednotlivá rizika a jejich hodnocení.

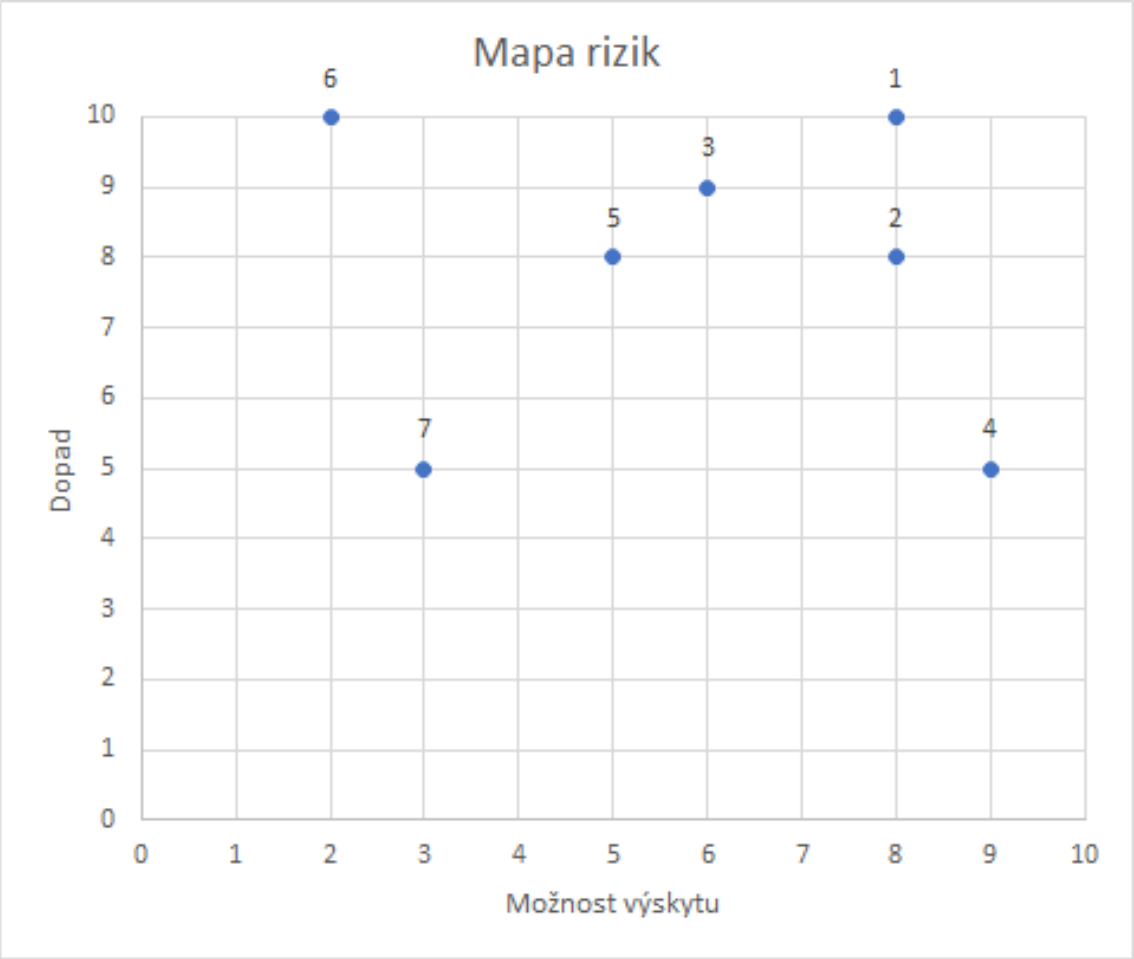
ID	Riziko	V	D	H
1	Odchod zákazníků kvůli špatnému technickému stavu systému	8	10	80
2	Snižování efektivnosti týmu a zvýšení nákladů na vývoj	8	8	64
3	Snížení prodejů kvůli pomalému načítání stránek	6	9	54
4	Další zpomalování IS vytvářením nových eshopů	9	5	45
5	Výpadky systému v důsledku přetížení	5	8	40
6	Ztráta dat kvůli pádům serverů	2	10	20
7	Penalizace vyhledávači za pomalé načítání	3	5	15

Tabulka 2.2: Analýza rizik (V = možnost výskytu, D = dopad, H = hodnota rizika)

Na základě tabulky jsem sestrojil mapu rizik, kterou znázorňuje graf 2.12. Čísla u jednotlivých bodů značí ID rizika.

Je patrné, že rizika 1, 2 a 3 jsou v kvadrantu kritických hodnot a je tedy nutné se na ně prioritně zaměřit a navrhnout vhodná opatření. Jejich stručný výčet shrnuje tabulka 2.3, detailněji se jimi zabývám v kapitole 3.

Zmíněná opatření by měla sekundárně pokrýt i ostatní, nekritická rizika, což představuje v tomto ohledu velkou výhodu a úsporu času a prostředků.



Obrázek 2.12: Mapa rizik současného stavu

Riziko	**Opatření**
1, 3	Změna architektury databáze Rozvoj strojových testů
2	Zavedení projektového řízení Zlepšení vývojového workflow

Tabulka 2.3: Opatření rizik

Kapitola 3

Vlastní návrhy řešení

V této kapitole se budu věnovat definici požadavků na zlepšení stávajícího stavu, sestavím Lewinův model řízení změn včetně mých návrhů na změny a budu prezentovat dosažené výsledky po aplikování navrhovaných úprav.

3.1 Požadavky na zlepšení

Na základě stávajícího stavu a následné diskuze mezi vývojovým týmem a majitelem služby byly sestaveny následující požadavky na zlepšení celkového stavu:

1. Zlepšení odezvy systému
2. Zvýšení jeho stability
3. Snížení šance vzniku regresí (chyb v důsledku implementace nových funkcí či přepracování těch stávajících)
4. Snížení nákladů na provoz a vývoj systému
5. S ohledem na malou velikost týmu zavedení alespoň základních principů projektového řízení

3.2 Lewinův model řízení změn

S ohledem na provedenou analýzu stávajícího stavu a požadavky na změny sestavím Lewinův model pro řízení těchto změn.

3.2.1 Síly působící na změnu

Součástí Lewinova modelu je sestavení seznamu sil působících ve prospěch, resp. proti provedení změn a kvantifikace těchto sil. K tomuto účelu jsem sestrojil tabulku 3.1. Váhu jednotlivých sil jsem stanovil v rozmezí -10–10.

Síly působící pro změnu		**Síly působící proti změně**	
Zrychlení odezvy systému	7	Programátoři	-3
Zvýšení stability systému	9	Možné technické komplikace	-5
Snížení šance vzniku regresí	8		
Zlepšení efektivity týmu	5		
Celkem pro	**29**	**Celkem proti**	**-8**

Tabulka 3.1: Síly působící na změnu

Z tabulky je patrné, že značně převládají pozitivní síly ve prospěch provedení změn. Programátoři nejsou zásadně proti navrženým změnám, nicméně mají obavy ze zásadních strukturálních úprav informačního systému.

3.2.2 Agent změny

Agentem změny, neboli zodpovědnou osobou za provedení navržených změn budu já jakožto vedoucí vývojářů. O průběhu prováděných změn budu informovat sponzora změny, kterým je majitel služby.

3.2.3 Intervenční oblasti

Navržené změny se dotknou v podstatě všech oblastí a zainteresovaných stran. Úpravy informačního systému ovlivní jak naši společnost (zrychlení práce se systémem, snížení nákladů na provoz serverů aj.), tak i naše zákazníky (zlepšení odezvy eshopů, vyšší stabilita apod.). Ti budou o změnách předem informováni prostřednictvím systému novinek v administraci eshopu. Změny v projektovém řízení týmu ovlivní efektivnost práce celého týmu včetně majitele služby.

3.2.4 Vlastní provedení změny

Změny budou probíhat dle navrženého harmonogramu sestaveného pomocí síťového grafu typu PERT.

Fáze rozmrazení

Analytická část této fáze je již dokončena a věnuje se jí kapitola 2. Taktéž již bylo rozhodnuto ve prospěch provedení těchto změn. V této fázi je nutné informovat všechny zainteresované strany.

Fáze provádění změn

Detailní informace o samotných změnách popisují kapitoly 3.3, 3.4 a 3.5.

Fáze zamrazení

V této fázi proběhne kontrola provedených změn a případné řešení vzniklých komplikací.

3.2.5 Verifikace dosažených výsledků

Prezentaci dosažených výsledků a jejich zhodnocení se věnuje kapitola 3.8.

3.3 Zavedení projektového řízení

S ohledem na fakt, že tým čítá pouze 5 členů (z toho 3 vývojáře) a případné náklady se prozatím nevyplatí zavádět oficiální formy certifikace projektového řízení. Nicméně budou zavedeny projektové procesy pro zadávání, průběh, kontrolu a dokončování projektů, které se budou opírat o běžné principy a metodiky projektového řízení.

3.3.1 Platforma Trello

Služba Trello slouží k jednoduché organizaci týmů či jednotlivců, kteří mohou spolupracovat online na nejrůznějších projektech. Základní organizační strukturou je

nástěnka. Každý uživatel si může pro sebe nebo svůj tým vytvářet jednotlivé nástěnky (v placené verzi je počet neomezený). Každá nástěnka pak nabízí možnost vytvářet sloupce, a do nich organizovat karty. Jednotlivé sloupce se řadí vedle sebe a je možné jejich pořadí libovolně měnit, taktéž je možné karty přesouvat mezi libovolnými sloupci.

Hlavní informační strukturu představují právě karty – ty obsahují svůj název, volitelný popis a následně je možné ke kartám připínat různé doplňky – zainteresované uživatele, seznamy úkolů, termíny, přílohy a mnoho dalšího. Ke každé kartě je pak možnost vkládat komentáře. Každý uživatel může sledovat libovolnou kartu (a automaticky sleduje karty, na které je připnut), na základě sledovaných pak uživatelé dostávají upozornění o aktivitách. Celý tento koncept je ideální pro jednoduchou organizaci malých až středních týmů.

Nově bude sloužit pro řízení vývojových projektů výhradně platforma Trello a každý nový (podstatnější) úkol bude řízen jako projekt.

Koncepce služby Trello je ideální pro aplikaci metody Kanban (viz 1.8.3). Zavedou se tedy primárně 4 hlavní sloupce – **Drafts** (návrhy na projekty, které dosud nebyly analyzovány), **TODO** (projekty, které dosud nebyly zahájeny), **WIP** – **Work In Progress** (právě zpracovávané projekty) a **Done** (dokončené projekty). Dle principů metody Kanban také bude platiti, že **vývojář nesmí mít přiřazen více jak jeden projekt ve sloupci WIP**.

Využívání platformy Trello umožní týmu lépe a efektivněji spolupracovat, sledovat činnosti ostatních členů a rychleji řešit případné problémy či překážky v práci. Vizualizace celého procesu prostřednictvím sloupců, karet a zaškrtávacích polí by mohla zlepšit motivaci týmu a předejde problémům přetěžování jednotlivých zdrojů (programátorů). Taktéž nám umožní podpořit vývoj a implementaci navržených změn dále popisovaných v kapitole 3.4.

3.3.2 Proces zadávání projektů

Pokud se majitel služby rozhodne pro implementaci nových či úpravu stávajících funkcí systému, bude nyní postupovat podle následujícího procesu:

1. Majitel služby navrhne první verzi zadání úkolu, zveřejní ji na Trello jako kartu ve sloupci Drafts. Název karty bude stručným subjektem činnosti projektu – například *Implementace vlastního dropshippingu*, popis bude obsahovat detailní popis včetně případných ilustrací.
2. Vedoucí vývoje si návrh prostuduje a spolu s majitelem sestaví analýzu rizik projektu prostřednictvím skórovací metody (zmiňuji se o ní v kapitole 1.8.5) a na základě ní rozhodnou, jaká opatření je nutné přijmout pro potlačení těchto rizik.
3. Následně sestaví časovou analýzu pomocí PERT (vizte 1.8.6) a vyplývající odhad nákladů.
4. Na základě analýzy rizik (schopnosti je potlačit či odstranit), časové a nákladové analýzy, majitel služby rozhodne, zda bude projekt možné realizovat. V případě negativního výsledku je možné upravit zadání tak, aby těmito kritérii prošel a provést znovu první 2 body.
5. Po odsouhlasení projektu se do informací zapíší výsledky zmíněných analýz a projekt se přesune do sloupce TODO.
6. V případě, že je nějaký vývojář k dispozici (nezpracovává žádný projekt), bude mu úkol po domluvě přidělen, po konzultaci s vedoucím vývoje bude projekt přesunut do sloupce Work In Progress a přiřazený vývojář začne úkoly zpracovávat.

3.3.3 Proces zpracování projektů

V průběhu každého projektu se bude postupovat dle následujícího procesu:

1. Vývojář zodpovědný za úkol bude průběžně aktualizovat stav projektu prostřednictvím přiřazených zaškrtávacích úkolů – pokud nazná, že je nutné do úkolů přidat nové položky, prokonzultuje to s vedoucím vývoje.
2. V případě komplikací při zpracování vývojář kontaktuje vedoucího vývoje (a případně majitele služby), kteří se pokusí zábrany odstranit.

3. Po dokončení všech specifikovaných úkolů vývojář provede finální testování, ověří, že je veškerá nová funkcionalita pokryta jednotkovými testy, a následně o dokončení informuje vedoucího vývojářů.
4. Vedoucí vývoje provede finální revizi kódu a kontrolu správného fungování řešení, v případě odhalených nedostatků doplní vývojáři potřebné úkoly a ten pokračuje opět bodem 1.
5. Pokud proběhnou všechny formy kontrol v pořádku, projekt je přesunut do sloupce Done a vedoucí vývojářů po dohodě provede nasazení nových funkcí do ostrého provozu.

3.3.4 Komunikace

Služba Trello umožňuje psát ke každé kartě komentáře, což se jeví jako vhodný prostředek komunikace vztahující se k samotným projektům – všichni zainteresovaní členové jsou informováni o nové aktivitě prostřednictvím notifikací.

V případě obecných informací může nadále být využívána forma emailu, tyto by však vždy měly být rozesílány celému týmu a nespoléhat se na interní komunikaci mezi členy týmu.

Pro rychlou komunikaci jsme také zřídili týmový chat na Messengeru, kde majitel a vývojáři mohou mezi sebou komunikovat. Pro komunikaci používáme *zmínky*, což je označení jména člověka, kterému je zpráva určena a s tím související nastavení, které upozorňuje pouze na zprávy, které zmiňují daného člena týmu.

Tato opatření zlepší **efektivnost komunikace týmu** a **eliminují nechtěný informační šum** pro členy, kterým není daná komunikace výhradně určena.

3.4 Změna architektury IS

Zásadním jádrem problému současného stavu, jak již bylo zmíněno v předešlých kapitolách, je technický stav informačního systému.

3.4.1 Změna architektury databáze

Změna architektury databáze by měla při vhodné implementaci pokrýt požadavky 1 – 3. Problémem databáze je její sdílená povaha a vznik duplicitních záznamů v lokálních tabulkách (produkty, kategorie, ...) – tabulky tak narůstají s každým novým eshopem a práce s nimi se značně zpomaluje, a vyžaduje velké množství klíčů, které naopak zpomalují operace vytváření/mazání/upravování.

Logickým řešením se tedy nabízí tyto tabulky rozdělit. Vytváření nových tabulek pro každý eshop ve stejné databázi by však z mnoha důvodů nebylo optimálním řešením – jak kvůli přehlednosti, tak kvůli nárokům interního systému MariaDB na skladování informací o všech tabulkách dané databáze. Jako nejlepší řešení se jeví vytvořit vlastní databázi pro každý eshop, která bude obsahovat lokální tabulky. V globální databázi pak zůstane nabídka produktů a dodavatelů, centrální evidence samotných eshopů, uživatelů, apod. Eshopové databáze se budou pojmenovávat podle konvence `shop_[ID]` (např. `shop_100000`), což umožní dynamicky upravit databázové dotazy – namísto příznaku `shop_id` ve `WHERE` části dotazů se všem lokálním tabulkám přidá prefix s názvem databáze – například tedy `shop_100000.product`.

Tabulky oddělené do vlastních databází nebudou vyžadovat sloupec `shop_id` ani stávající velké množství indexů, protože práce s nimi bude několikanásobně rychlejší i s použitím minima klíčů – každá tabulka lokálních produktů bude mít např. 50 000 záznamů namísto několika milionů. To povede ke zrychlení čtecích i zápisových operací.

Tato změna architektury bude vyžadovat zásadní úpravy v backendu informačního systému. Pro co nejplynulejší přechod byl navržen systém, který umožňuje v konfiguračním souboru IS namapovat všechny lokální eshopové tabulky pod danou databázi. Konfigurace využívá formát JSON a vypadá například takto:

```
"db_mappings": {
        "shop_{id}": [
                "product",
                "margin",
                "image",
                "order",
                "order_basket",
                "basket",
                "variant",
                "shop_supplier",
                "shop_module",
                "category"
        ]
}
```

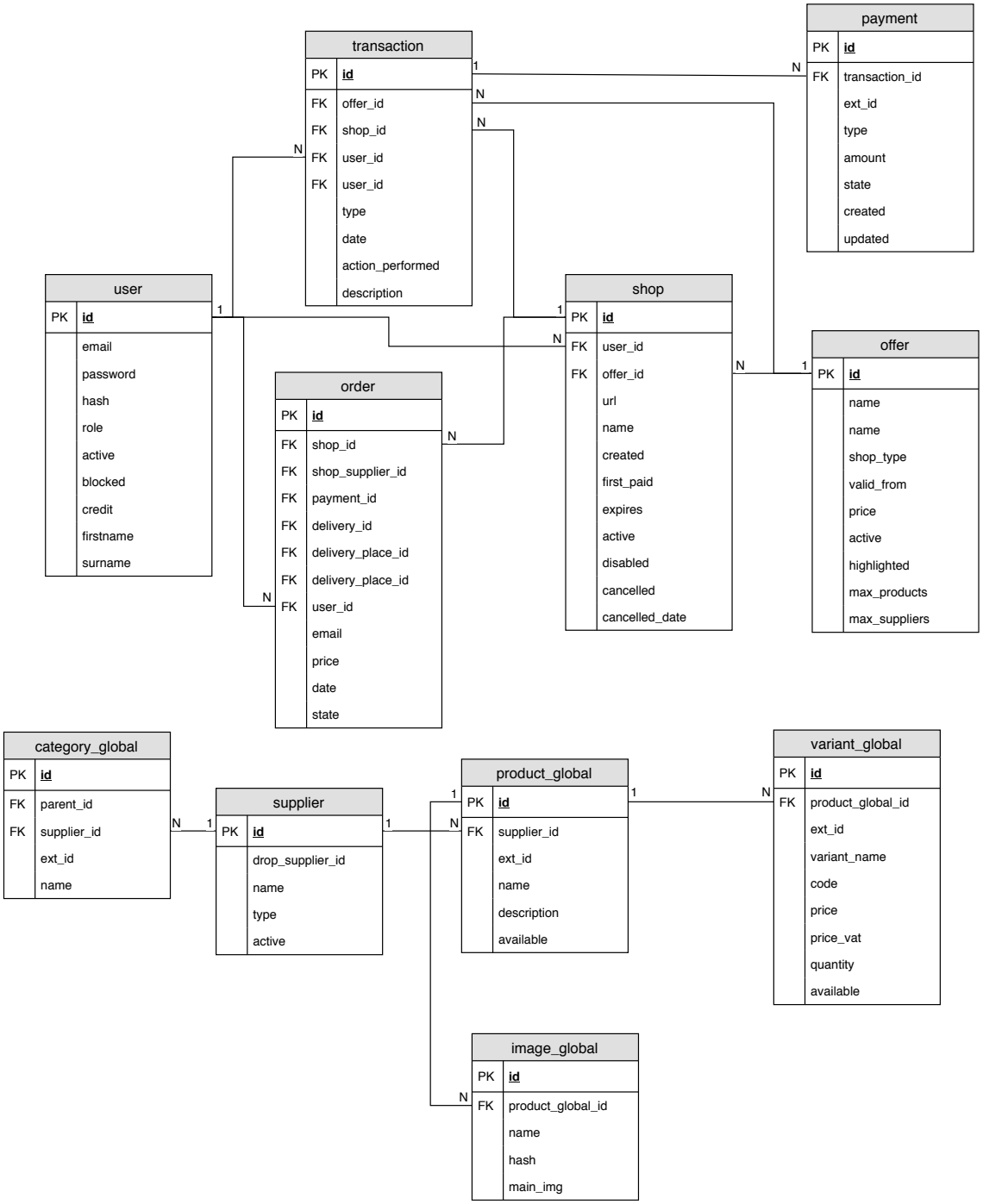
Výpis 3.1: Ukázka konfigurace mapování

Proměnná `{id}` v názvu databáze bude dynamicky nahrazována ID právě používaného eshopu. V systému se nicméně nachází místa, kde bude nutné provést ruční změnu (například u pod dotazů, které nepoužívají dynamický generátor SQL dotazů) – toto bude vyžadovat hromadné prohledání všech skriptů a postupné nahrazení s kontrolou.

Stav databáze po změně shrnují ER diagramy 3.1 (globální databáze) a 3.2 (lokální, eshopová databáze).

Navržené změny by měly přinést především **škálovatelnost** – systém se již nebude zpomalovat úměrně k počtu nových eshopů a nebudou vznikat zbytečné duplicity v rámci lokálních tabulek, protože každý eshop bude mít svá vlastní data.

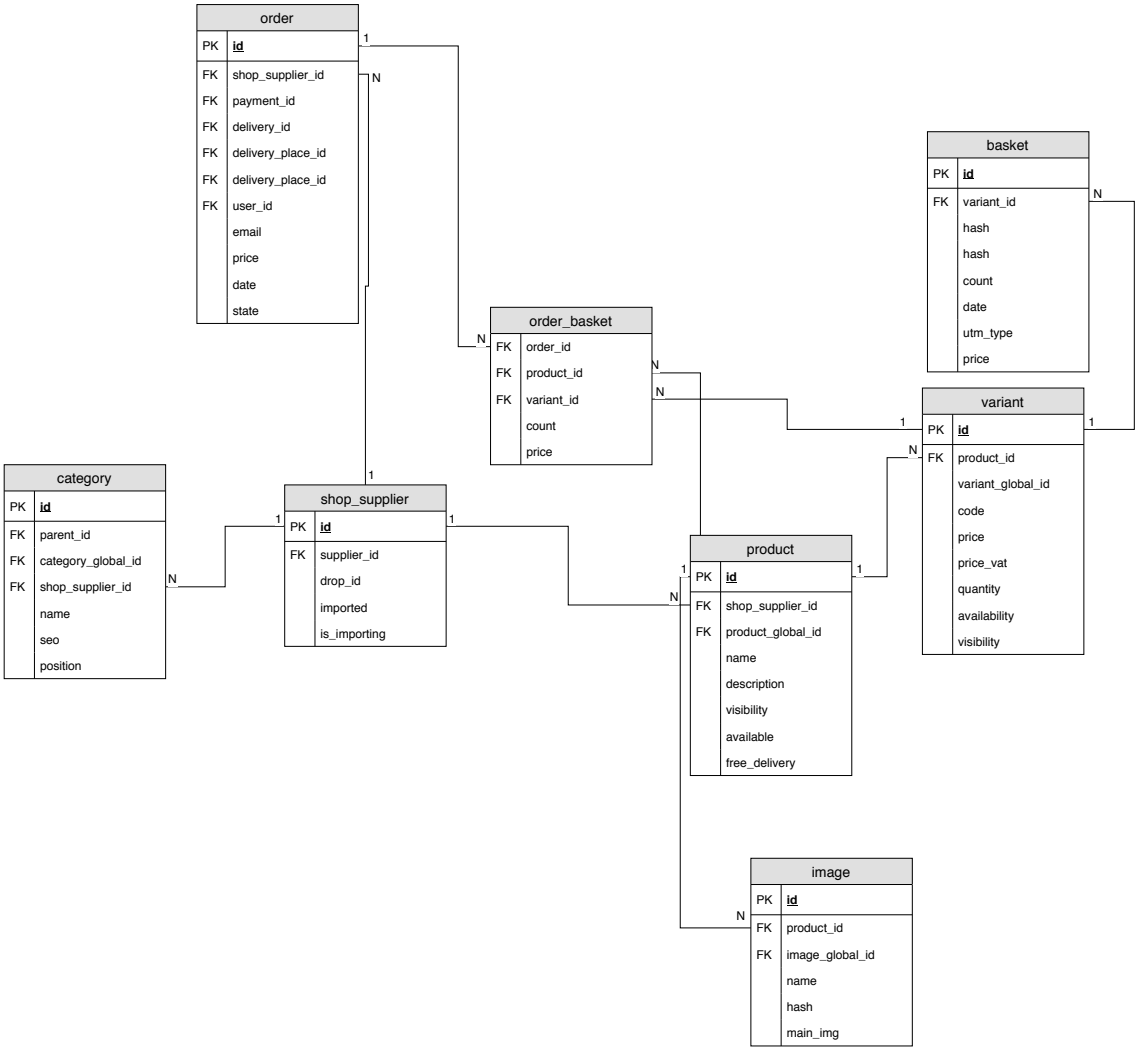
To přinese **mnohem lepší odezvu systému**, jelikož se budou procházet tabulky pouze o velikosti, která odpovídá počtu dat daného eshopu – například se bude vyhledávat v tabulce 10 000 produktů namísto několika milionů. Tím pádem se bude moci také **snížit počet indexů**, které zpomalují zápisové operace.



Obrázek 3.1: ER diagram upraveného schématu globální databáze

3.4.2 Zlepšení strojových testů

V rámci zlepšení celkového stavu provedu zvýšení pokrytí jednotkovými testy. S ohledem na velké množství metod a komplexnost celého systému považuji za reálné navýšit pokrytí metod maximálně na 30 %.



Obrázek 3.2: ER diagram upraveného schématu lokální (eshopové) databáze

Bude nutné primárně pokrýt kritické funkce, o které se opírá velká část systému nebo které přímo ovlivňují provoz služby a tržeb. Navýšení tohoto pokrytí zajistí **snížení rizik** vyplývajících z implementace nových funkcí či úpravy těch stávajících. Zvýší to celkovou efektivitu vývoje a sníží frustraci a nejistotu vývojářů při změně důležitých součástí systému.

3.5 Zlepšení vývojového workflow

Pro naše vývojové workflow je nezbytné zavést několik vylepšení a opatření, jejich výčet jsem shrnul v seznamu níže.

- Pro vývoj každé nové funkcionality bude vytvořena nová větev pojmenovaná ve stylu `feature/strucny_nazev`, například tedy `feature/credit_cards`.
- Pro změnu stávající funkcionality bude vytvořena nová větev ve stylu `update/strucny_nazev` například tedy `update/xml_feeds`.
- Pro opravy budou vytvořeny větve pojmenované `fix/strucny_nazev`, takže například `fix/mobile_design`.
- Zápisový přístup k hlavní větvi `master` bude mít pouze vedoucí vývojářů.
- Bude vytvořena nová sekundární větev `dev`, která bude sloužit jako akumulátor nových funkcí a změn, aby bylo možné provést integrační testování více samostatných změn, zápisová oprávnění bude mít opět pouze vedoucí vývojářů.
- Pokud budou chtít vývojáři provést operaci merge s větví `master` nebo `dev`, musí nově vytvořit pull request (žádost o sloučení).
- Bude zavedena automatizovaná průběžná integrace (CI) v podobě našich strojových testů a kontroly kvality kódu.
- Každý pull request bude podroben automatizovanému testování pomocí CI nástroje.
- Větev `master` i `dev` budou průběžně podrobovány automatizovanému testování.
- Vedoucí vývojářů bude pull requests schvalovat na základě výsledků automatizovaných testů a odborného posudku.

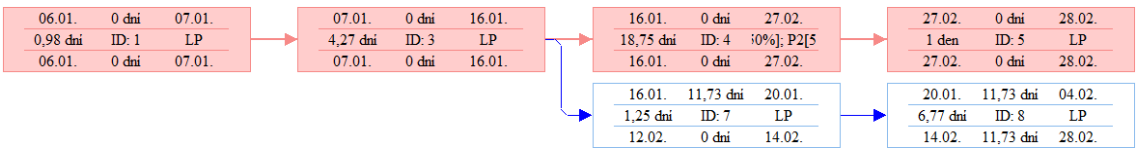
Tento seznam změn by měl docílit **snížení rizika proniknutí chyb do ostrého provozu** a **zvýšení efektivity a robustnosti celého vývojového procesu**. Taktéž bude motivovat vývojáře průběžně vyvíjet jednotkové testy spolu se samotným vývojem nových funkcí.

3.6 Časová a nákladová analýza

Na základě zmíněných návrhů jsem sestavil časovou (tabulka 3.2) a nákladovou (tabulka 3.3) analýzu. K vytvoření časové analýzy byla použita metoda PERT a byla vytvořena pomocí programu Microsoft Project. Z této časové analýzy byla sestavena i nákladová analýza. Zdroje jsou označeny následovně: LP (lead programmer – vedoucí vývojář), P1 a P2 (programátoři). Nákladová analýza počítá s platem vedoucího programátorů 450 Kč na hodinu a vývojářů 350 Kč na hodinu. Nákladová analýza vychází pouze z variabilních nákladů za práci, nefigurovaly zde žádné fixní náklady. Pro lepší ilustraci slouží PERT diagram na obrázku 3.3.

ID	**Úkol**	$a_{i,j}$ **[člh]**	$m_{i,j}$ **[člh]**	$b_{i,j}$ **[člh]**	$t_{i,j}$ **[čld]**
1	Zavedení procesů pro řízení projektů pomocí Trello	5	8	10	0,98
3	Příprava zadání úprav databáze a systému, základní implementace	25	35	40	4,27
4	Samotné provedení změn	120	150	180	18,75
5	Nasazení do ostrého provozu	6	8	10	1
7	Napojení a konfigurace průběžné integrace (CI)	8	10	12	1,25
8	Zlepšení pokrytí jednotkových testů	45	50	80	6,77
Celkem					**25**

Tabulka 3.2: Časová analýza, a – optimistický odhad, m – realistický odhad, b – pesimistický odhad, t – předpokládaný čas



Obrázek 3.3: PERT diagram

ID	Úkol	Náklady [Kč]
1	Zavedení procesů pro řízení projektů pomocí Trello	3 528
3	Příprava zadání úprav databáze a systému, základní implementace	15 372
4	Samotné provedení změn	52 500
5	Nasazení do ostrého provozu	3 600
7	Napojení a konfigurace průběžné integrace (CI)	4 500
8	Zlepšení pokrytí jednotkových testů	24 372
Celkem		**103 872**

Tabulka 3.3: Nákladová analýza

3.7 Analýza rizik změn

Vzhledem k rozsahu navržených změn a celkovému současnému stavu je více než žádoucí provést analýzu rizik provedení těchto změn, stejným způsobem, jakým byla provedena před změnami (2.6).

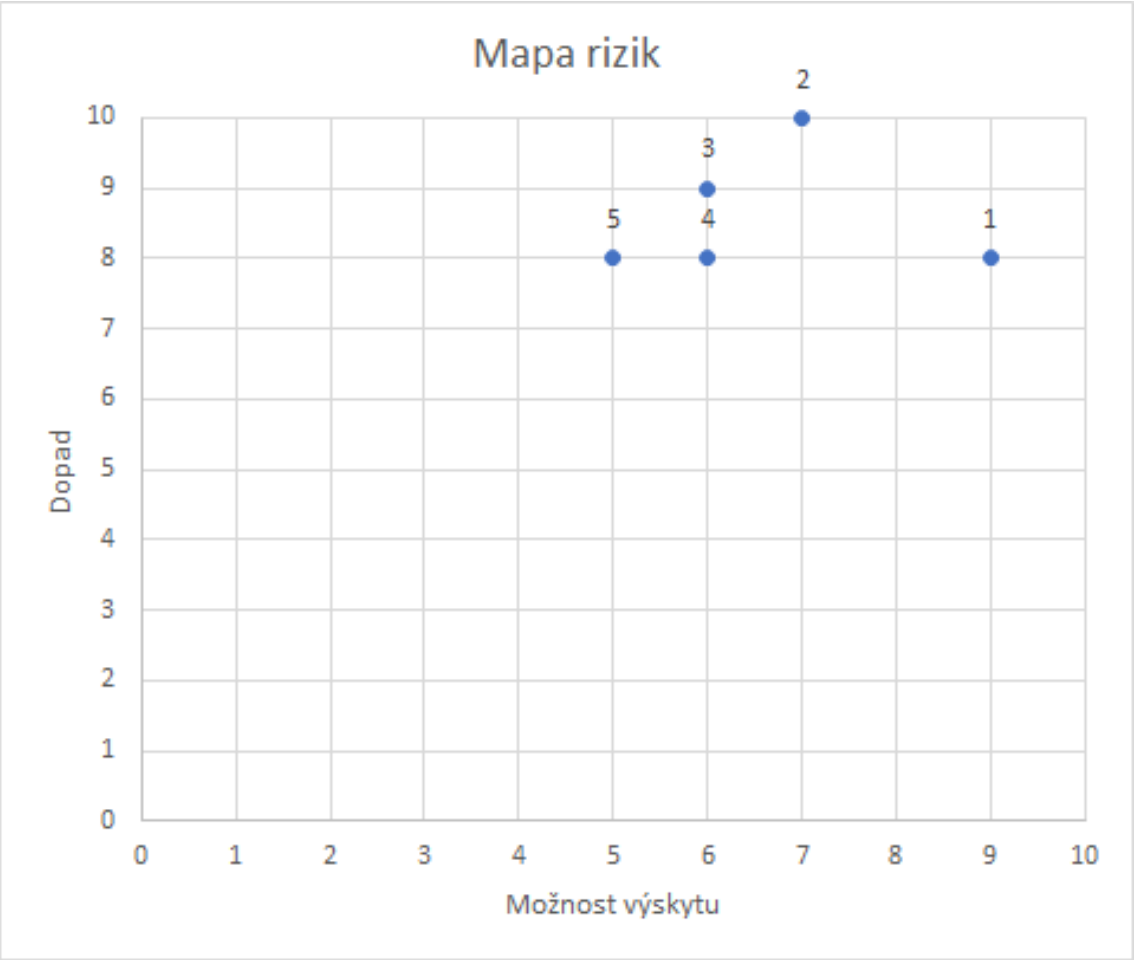
Nyní zanalyzuji rizika navržených změn pomocí skórovací metody, souhrn mé analýzy je zachycen tabulkou 3.4.

ID	Riziko	V	D	H
1	Vznik chyb IS v důsledku rozsáhlých změn	9	8	72
2	Ztráta či poškození integrity dat při migraci databázové architektury	7	10	70
3	Vysoké náklady na dostatečné pokrytí strojových testů	6	9	54
4	Technické komplikace ze strany serverů či hardwaru	6	8	48
5	Špatná adaptace principů projektového řízení	5	8	40

Tabulka 3.4: Analýza rizik změn (V = možnost výskytu, D = dopad, H = hodnota rizika)

Na základě této tabulky jsem opět sestrojil mapu rizik vyobrazenou grafem 3.4.

Z grafu je patrné, že rizika 1, 2, 3 a 4 se nachází v kvadrantu kritických hodnot. na základě tohoto zjištění jsem sestavil tabulku opatření 3.5.



Obrázek 3.4: Mapa rizik navržených změn

Riziko	**Opatření**
1	Průběžně testovat funkcionalitu, zavádět strojové testy rovnou při aplikaci změn
2	Provedení zálohy dat, neodstraňovat původní architekturu z databáze, dokud nebude jisté, že nový systém funguje
3	Testovat především kritické a netriviální části systému
4	Příprava a záloha serverů před provedením změn, kontrola nastavení

Tabulka 3.5: Opatření rizik změn

3.8 Dosažené výsledky

Vzhledem k tomu, že navržené změny byly již úspěšně implementovány, představím zde výsledky a zlepšení, kterých bylo mými návrhy dosaženo.

3.8.1 Odezva systému

Stejně jako v analytické kapitole 2.2.5 jsem změřil odezvu systému na stejných případech. Byla dodržena totožná testovací metodika a testy byly prováděny zhruba ve stejnou denní dobu jako předchozí testy pro zachování konzistence výsledků.

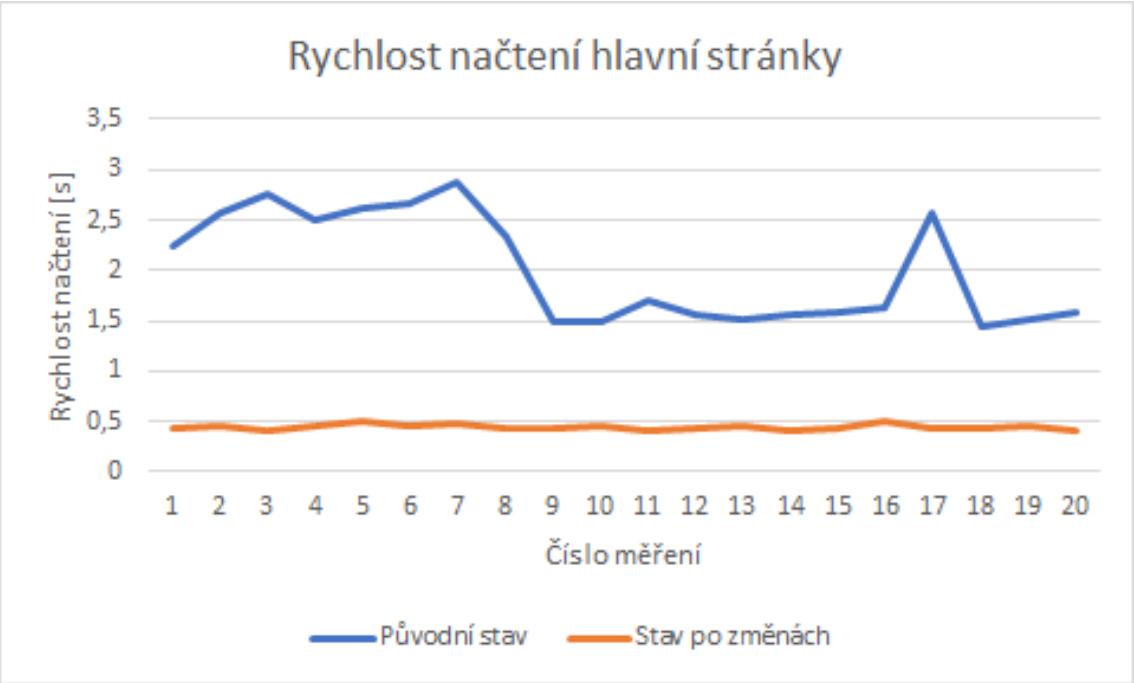
Rychlost načítání eshopu

Z výsledků měření po implementaci mnou navržených změn jsem sestavil tabulku 3.6, která shrnuje dosažená zlepšení a odkazuje na pomocné grafy.

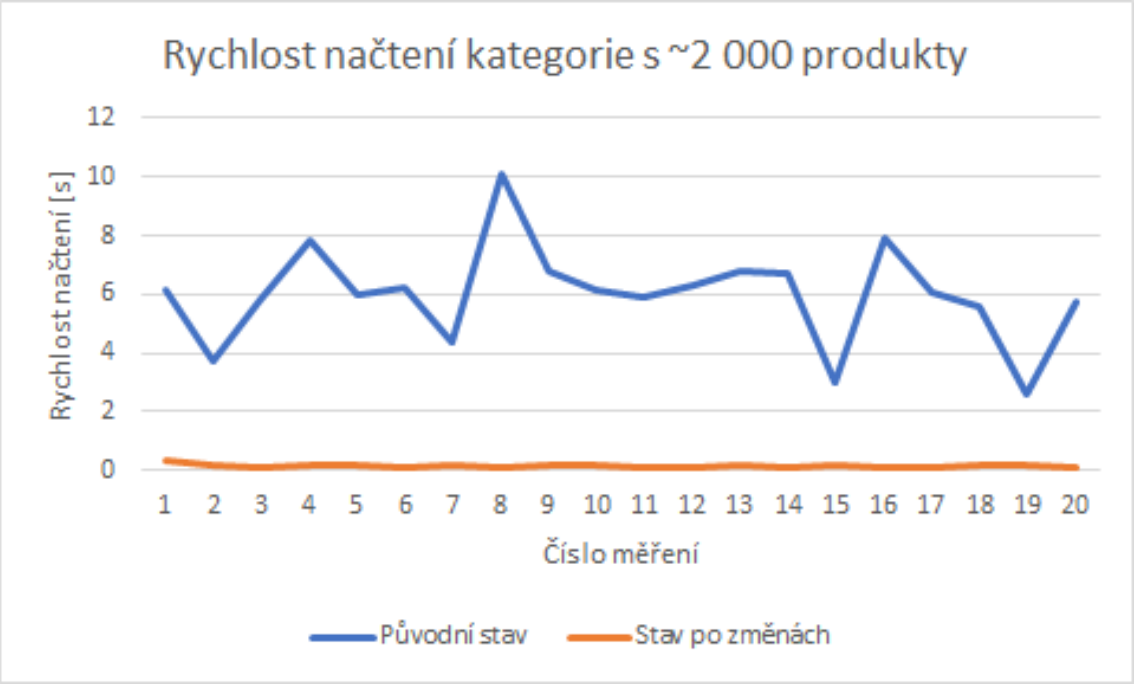
Měření	Min. zrychlení	Prům. zrychlení	Max. zrychlení	Graf
Rychlost načtení hlavní stránky	2,88× (65,31 %)	4,55× (78,02 %)	7,13× (85,97 %)	3.5
Rychlost načtení kategorie	7,72× (87,04 %)	39,24× (97,45 %)	78,6× (98,73 %)	3.6
Rychlost načtení detailu produktu	0,57× (-75,47 %)	1,66× (39,71 %)	7,14× (86 %)	3.7

Tabulka 3.6: Výsledky odezvy systému

Z tabulky a přiložených grafů vyplývá, že často došlo až k drastickému zrychlení odezvy systému. Z křivek na grafech také vyplývá, že odezva je po změnách mnohem stabilnější a nevyskytují se zde téměř žádné směrodatné výkyvy. To pomůže k rychlejšímu zpracování požadavků a značně nižší zátěži na všech serverech.



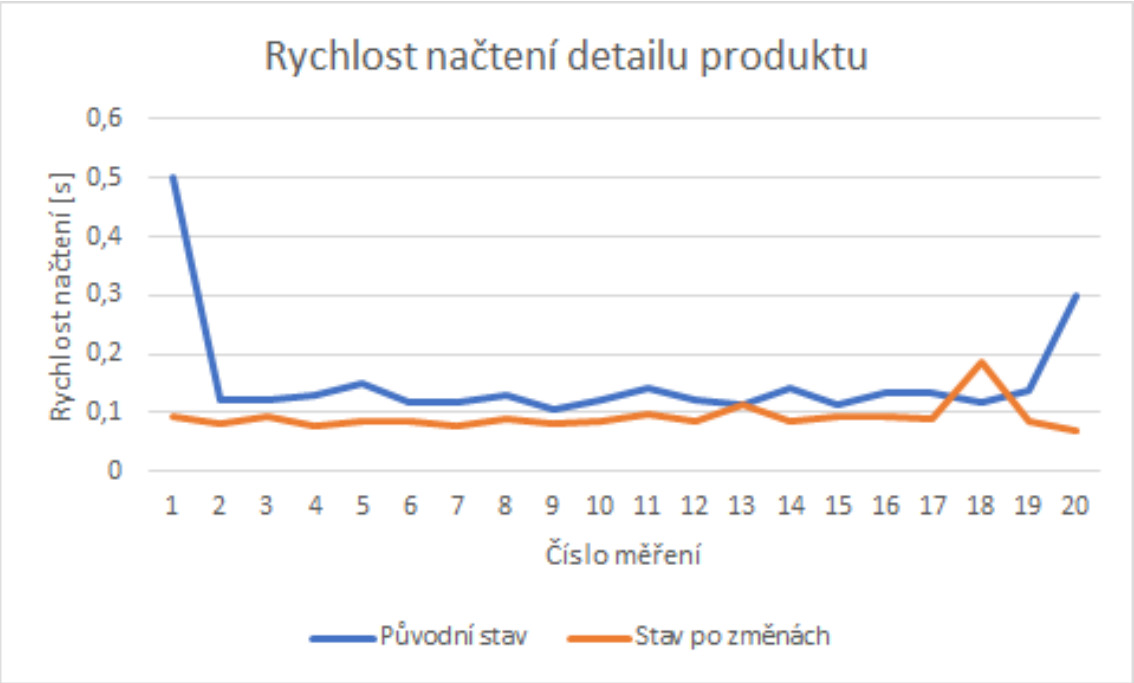
Obrázek 3.5: Srovnání rychlosti načtení hlavní stránky před a po změnách



Obrázek 3.6: Srovnání rychlosti načtení kategorie před a po změnách

Rychlost generování XML feedů

V rámci testování dopadů provedených úprav jsem taktéž opět změřil rychlost generování XML feedů pro cenové srovnavače, zlepšení znázorňuje tabulka 3.7 a graf 3.8.

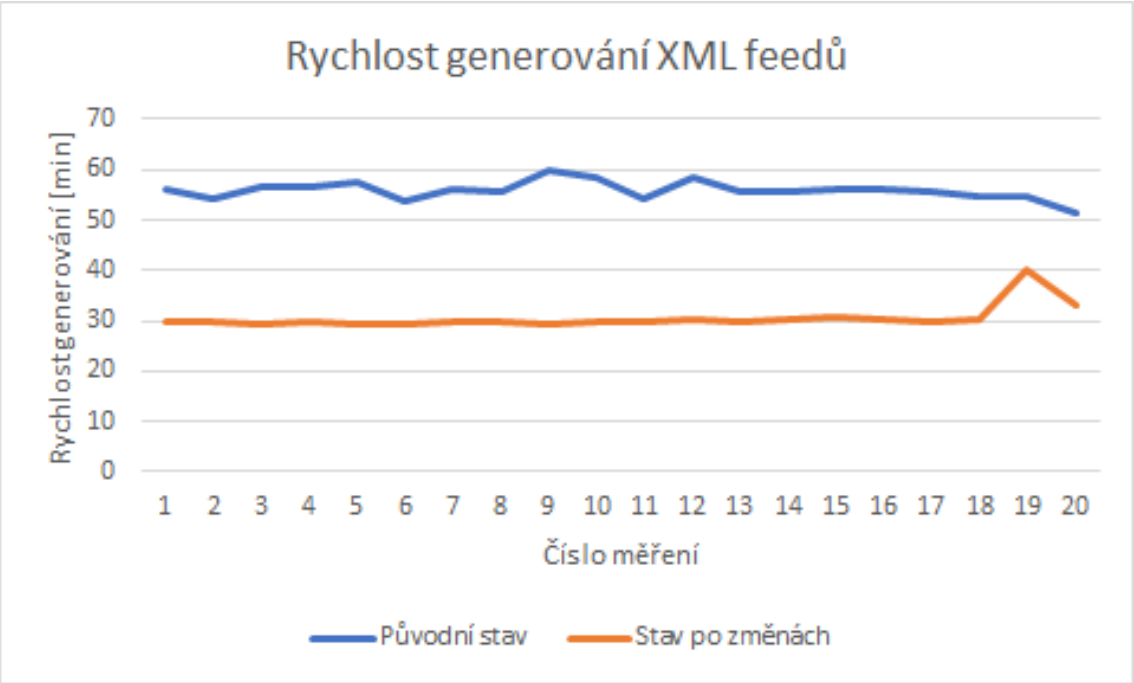


Obrázek 3.7: Srovnání rychlosti načtení detailu produktu před a po změnách

Měření	**Min. zrychlení**	**Prům. zrychlení**	**Max. zrychlení**
Rychlost generování XML feedů	1,29× (22,19 %)	1,83× (45,47 %)	2,04× (51,04 %)

Tabulka 3.7: Výsledky generování XML feedů

Z tabulky a grafu je patrné, že bylo dosaženo přibližně dvojnásobného zrychlení bez jakýchkoliv dalších úprav samotného generovacího algoritmu – to v budoucnu například umožní generovat XML feedy častěji, takže budou obsahovat aktuálnější data o produktech. Lze také vypozorovat, že křivka nového stavu má minimální odchylky – čas generování je tedy podstatně konzistentnější.



Obrázek 3.8: Srovnání rychlosti generování XML feedů před a po změnách

3.8.2 Stav strojových testů

V rámci strojových testů bylo vynaloženo značné úsilí na pokrytí alespoň kritických funkcí, které mají velkou míru používání ve zbytku systému, či které přímo ovlivňují provoz IS a jeho kritických operací, jako jsou platební transakce apod.

Metrika	Pokrytí	Poznámka
Třídy	8,37 %	Udává, kolik tříd jazyka je kompletně pokryto testy
Metody	27,82 %	Udává, kolik metod jazyka je kompletně pokryto testy
Řádky kódu	43,11 %	Udává, kolik řádků kódu je kompletně pokryto testy

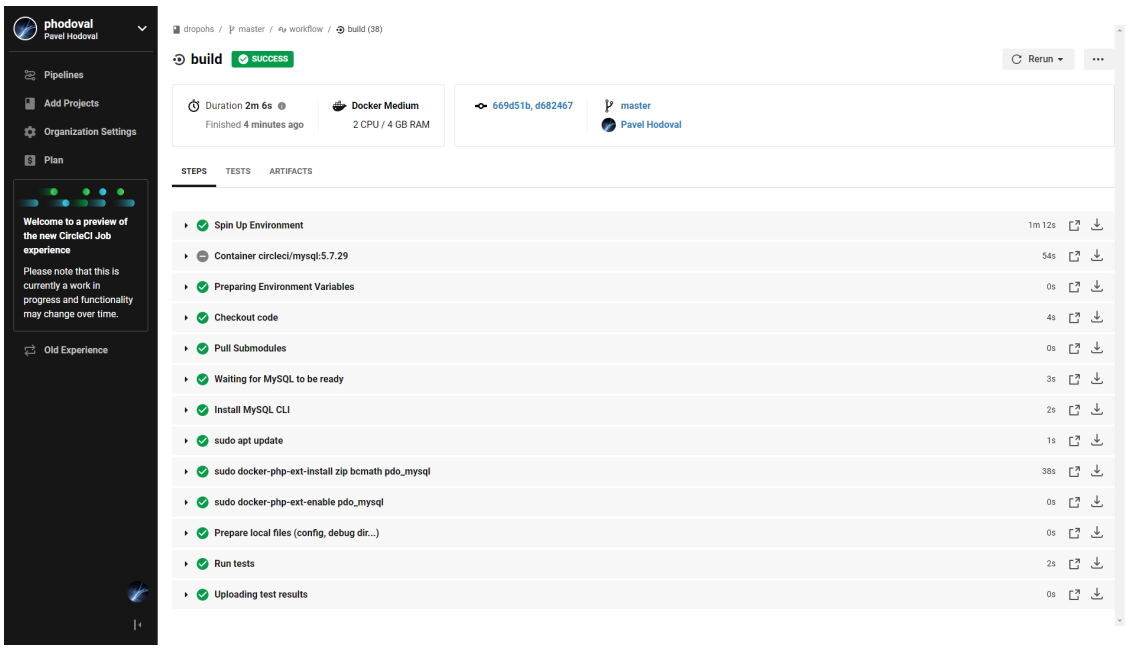
Tabulka 3.8: Pokrytí jednotkovými testy

Dosažené pokrytí je sice nižší, než bylo plánováno, ale podařilo se pokrýt většinu kritických součástí systému – mělo by to značně **snížit rizika vzniku chyb** při dalším vývoji a celkově zlepšit robustnost a stabilitu systému v rámci dalšího rozvoje.

3.8.3 Zlepšení vývojového workflow

Na základě návrhů v kapitole 3.5 jsem na našem GitHub repozitáři implementoval průběžnou integraci od CircleCI. Integrační úloha v současné době po nahrání každého nového commitu automaticky provede naše jednotkové testy a uloží výsledky do CircleCI. Taktéž zašle notifikaci přímo na GitHub, která zajišťuje, že u každého commitu se nachází informace o úspěšnosti testů.

Všichni vývojáři si pak mohou zobrazit detailní výsledky testů přímo v administraci systému CircleCI (obrázek 3.9). Nasazení této změny **ušetřilo vývojářům značné množství času**, jelikož již nemusí testy spouštět ručně. To vedlo i k celkovému snížení nákladů na vývoj. Taktéž to zjednodušilo revizi kódu vedoucímu vývoje, který má rychlý přehled o funkčnosti nahraných úprav.



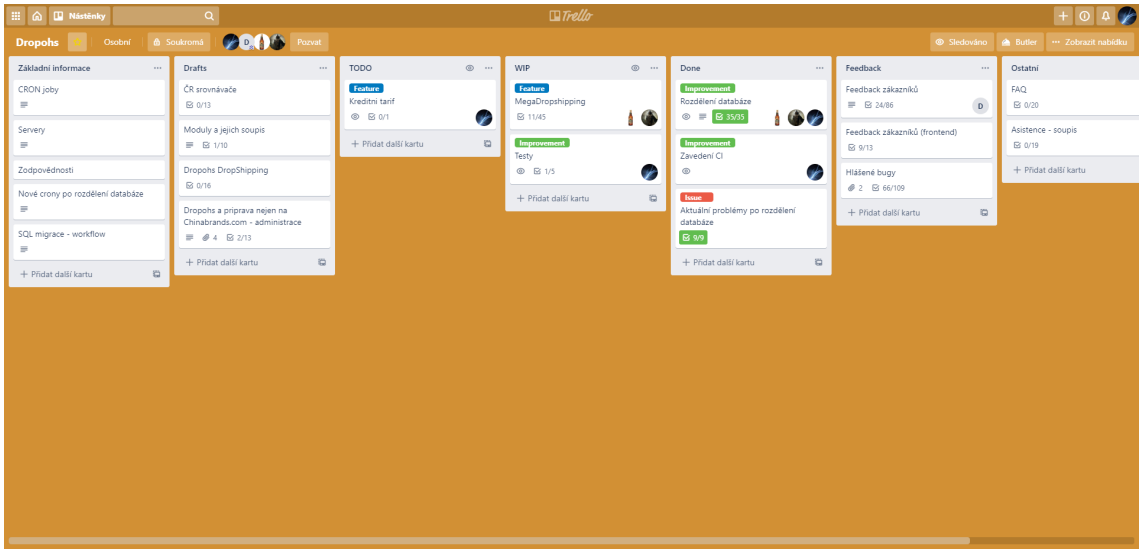
Obrázek 3.9: Ukázka rozhraní úlohy v CircleCI

Dále byla zřízena sekundární větev `dev`, která slouží k integraci změn, které dosud nebyly spuštěny do ostrého provozu. Na větvích `master` i `dev` byla upravena oprávnění tak, aby zde mohl publikovat změny pouze vedoucí vývoje – ostatní vývojáři musí nově založit pull request pro své úpravy. Průběžná integrace navíc každý pull request rovnou zkontroluje, takže se do *ostrých* větví dostanou pouze změny ověřené jak jednotkovými testy, tak vedoucím vývojářů.

3.8.4 Zavedení projektového řízení

Dle stanovených návrhů v kapitole 3.3 jsem implementoval a korektně přeorganizoval naši firemní Trello nástěnku. Úkoly byly rozděleny do příslušných sloupců, nesouvisející údaje byly přesunuty do jiných a byly zavedeny štítky pro lepší organizaci. Byla zavedena navrhovaná metoda Kanban pro správu úkolů (projektů) a minimalizaci přetěžování. Současnou organizační strukturu shrnuje obrázek 3.10.

Dále byl zaveden nový, dříve zmiňovaný, proces zadávání a řízení průběhu úkolů, který nejdříve putuje do sloupce Drafts, kde proběhnou potřebné diskuze, analýzy a závěry. Až poté se projekt přesouvá do sloupce TODO, kde čeká na zpracování vývojáři.



Obrázek 3.10: Ukázka struktury Trello

Členové týmu včetně majitele služby již dříve s nástrojem Trello pracovali, takže pro ně byla aklimatizace velmi rychlá. Zlepšila se také týmová komunikace – emaily jsou nyní rozesílány všem členům týmu, komunikace ke konkrétním úkolům je řešena prostřednictvím komentářů na Trello a k rychlé komunikaci se používá týmový chat na Messsengeru.

Provedené změny přinesly **mnohem lepší přehlednost aktuálního stavu prováděné práce** a **efektivnější organizaci týmu**. Vizuální stránka věci navíc podněcuje lepší motivaci vývojářů dokončovat své úkoly. Nasazené změny navíc **eliminovaly či odstranily mnohá rizika** – především přetěžování zdrojů, kon-

flikty v úkolech, špatnou informovanost a komunikaci v týmu, provádění rizikových projektů bez předešlé analýzy a podobně.

3.8.5 Odhalené nedostatky řešení

Po provedení navrhovaných změn informačního systému a jeho architektury byly odhaleny některé technické nedostatky a komplikace, které se projevily ihned nebo chvíli po nasazení do ostrého provozu.

S ohledem na novou architekturu databází, kde má každý eshop svou vlastní s několika tabulkami vzrostl celkový počet databázových tabulek na několik tisíc. To vedlo k chybám relační databáze, která neměla přidělenou dostatečnou velikost cache pro počet otevřených a definovaných tabulek (`table_open_cache` a `table_definition_cache`). Navýšením zmíněných hodnot byl problém odstraněn.

Dalším problémem způsobeným velkým množstvím tabulek se staly cizí klíče – tisíce tabulek z eshopových databází odkazovaly do globální databáze pomocí cizích klíčů, což vedlo k drastickému zhoršení výkonu zápisových operací nad globální databází. Příkladem je třeba lokální tabulka produktů `product`, která měla cizí klíč do tabulky `product_global`, aby bylo možné aktualizovat lokální produkty dle aktuálního stavu dodavatelů apod. Pro vyřešení problému bylo nutné odstranit všechny cizí klíče, které vedly z lokálních tabulek do globálních a zavést aplikační mechanismy pro udržení konzistence databáze. Cizí klíče vedoucí do stejné databáze nebylo nutné odstraňovat.

3.8.6 Zhodnocení výsledků

K datu vydání této diplomové práce byly všechny navržené změny úspěšně implementovány a nasazeny do ostrého provozu. Díky tomu můžeme provést reálné časové a ekonomické zhodnocení oproti odhadovaným hodnotám. Získaná data vychází z aplikace Toggl, kterou si vývojáři měří svou dobu práce a jejich průměrné hodinové sazby. Zjištěné výsledky shrnuje tabulka 3.9.

Z tabulky je patrné, že požadovaných cílů bylo dosaženo **rychleji a za menší náklady**, než bylo odhadováno. Odchylka od původního odhadu je způsobena hlavně

Položka	Původní odhad	Reálná hodnota
Časová náročnost	25 čld	20,88 čld
Náklady	103 872 Kč	86 964 Kč

Tabulka 3.9: Finanční a nákladové srovnání

urychlením práce díky implementaci částečné automatizace změn na úrovni backendu IS, který se díky konfiguraci globálně staral o změnu adresování databázových tabulek, čímž se snížila nutnost manuálního zásahu programátora, který pouze provedl rekonfiguraci dané struktury.

Od nasazení navrhovaných změn také dle serverového specialisty evidujeme **rekordně nejnižší zátěž serverů** od spuštění služby, což považuji za velký úspěch. Odezva systému se také razantně zlepšila, jak již bylo popisováno dříve. Celkové snížení náročnosti a odezvy má za následek **eliminaci či potlačení rizik** spojených s provozem informačního systému a přinesl také **snížení nákladů na provoz**.

Závěr

Cílem práce bylo provést analýzu současného stavu informačního systému pro vytváření eshopů na modelu dropshipping a efektivity práce vývojového týmu a na základě toho navrhnout a implementovat případná vylepšení a opatření za pomoci vybraných principů projektového řízení pro zefektivnění práce týmu.

Začátek práce se věnoval seznámení s nezbytnou teoretickou problematikou, ze které následně vycházím. Posléze byla provedena analýza současného stavu informačního systému, stavu vývojového workflow a projektového řízení v týmu.

Analýzou současného stavu informačního systému byly zjištěny nedostatky především v již nedostačující databázové architektuře, která zhoršovala s růstem systému odezvu i stabilitu. Dále bylo zjištěno, že pokrytí strojových jednotkových testů je ve velmi špatném stavu. V rámci vývojářského týmu byly nalezeny problémy v nedostatečném nastavení pravidel a automatizace v rámci vývojového workflow prostřednictvím nástroje Git a absence principů projektového řízení, což vedlo ke snižování efektivity týmu. Na základě současného stavu byla provedena analýza rizik a navržena opatření.

S ohledem na výsledky analýzy a sestavené požadavky bylo pomocí Lewinova modelu řízení změn navrženo několik úprav či vylepšení, která by měla zjištěná rizika odstranit či minimalizovat.

V rámci informačního systému bylo navrženo vytvářet pro každý nový eshop vlastní databázi a oddělit tak data eshopů od globálních dat systému a backendové úpravy pro podporu této změny.

U vývojového workflow jsem navrhl zavést systém oprávnění pro přístup k jednotlivým vývojovým větvím, používat mechanismus pull request, stanovení pravidel pro vytváření a práci s větvemi a zavedení průběžné integrace.

Pro tým jsem doporučil využít službu Trello, která umožňuje přehlednou organizaci práce do sloupců a karet a na základě toho využít metodu Kanban. Dále jsem sestavil nové procesy pro zadávání a řízení životního cyklu úkolů, které se nově měly řídit jako projekty. Součástí těchto procesů byla skórovací analýza rizik a metoda PERT pro časovou a nákladovou analýzu.

Následně byly všechny navržené změny implementovány, otestovány a byly srovnány dosažené výsledky. V rámci odezvy a stability informačního systému bylo dosaženo **několikanásobného zlepšení** (v extrémních případech až 70násobné snížení odezvy). Tím byla také snížena zátěž serverů a s tím spojené **snížení provozních nákladů**.

U vývojového workflow jsem stanovil a implementoval nová pravidla pro práci s větvemi a zavedl jsem průběžnou integraci prostřednictvím CircleCI. Tyto změny vedly ke **snížení vývojových rizik** a díky automatizaci také ke **zrychlení a zefektivnění vývoje**.

Pro projektové řízení byla použita navržená služba Trello a zavedeny všechny doporučené procesy. Zlepšila se **organizovanost i efektivita** práce v týmu, týmová komunikace a došlo ke snížení konfliktů v rámci jednotlivých projektů. Díky aplikaci vizuálních metod organizace také došlo ke zlepšení motivace vývojářů.

S ohledem na prezentované výsledky byly stanovené cíle naplněny a pro službu Dropohs.cz tato diplomová práce představuje velký přínos.

Literatura

[1] Bruckner, T., Voříšek, J., Buchalcevová, A. a kolektiv. *Tvorba informačních systémů: Principy, metodiky, architektury*. Grada Publishing a.s., 2012. ISBN 9788024779027.

[2] Chaffer, J. a Swedberg, K. *Mistrovství v jQuery: [kompletní průvodce vývojáře]*. 1. vyd. Brno: Computer Press, 2013. ISBN 9788025141038.

[3] Duvall, P. *Continuous Integration*. Pearson Education, 2007. ISBN 9788131722916.

[4] Foundation, T. A. S. *About The Apache HTTP Server Project - The Apache HTTP Server Project* [online]. [cit. 2020-01-18]. Dostupné z: https://httpd.apache.org/ABOUT_APACHE.html.

[5] Hayes, M. a Youderian, A. *The Ultimate Guide to Dropshipping*. Lulu Publishing Services, 2013. ISBN 9781483401812.

[6] Hodoval, P. *Sada webových nástrojů pro Vizualizace geografických dat*. Brno, CZ, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/106501.

[7] Křivánek, M. *Dynamické vedení a řízení projektů: Systémovým myšlením k úspěšným projektům*. 1. Praha: Grada Publishing a.s., 2019. ISBN 8027126452.

[8] Klčová, H. a Sodomka, P. *Informační systémy v podnikové praxi*. 1. Praha: Computer Press, Albatros Media a.s., 2017. ISBN 9788025145432.

[9] Kniberg, H. a Skarin, M. *Kanban and Scrum - Making the Most of Both*. C4Media Incorporated, 2010. Enterprise software development series. ISBN 9780557138326.

[10] Kubíčková, L. a Rais, K. *Řízení změn ve firmách a jinỳch organizacích*. Grada Publishing a.s., 2012. ISBN 9788024775982.

[11] Laster, B. *Professional Git*. Wiley, 2016. Online access with DDA: Askews. ISBN 9781119284970.

[12] Laurenčík, M. *SQL: podrobný průvodce uživatele*. 1. Praha: Grada Publishing, 2018. ISBN 978-80-271-0774-2.

[13] Lazaris, L. *CSS okamžitě*. 1. vyd. Brno: Computer Press, 2014. ISBN 9788025141762.

[14] Molnár, Z. *Podnikové informační systémy*. ČVUT, 2009. ISBN 9788001043806.

[15] Pries, K. H. a Quigley, J. M. *Scrum Project Management*. CRC Press, 2010. ISBN 9781439825174.

[16] Písek, S. *HTML: začínáme programovat*. 4., aktualiz. vyd. Praha: Grada, 2014. ISBN 9788024750590.

[17] Žára, O. *JavaScript: programátorské techniky a webové technologie*. 1. vydání. Brno: Computer Press, 2015. ISBN 9788025145739.

[18] Schwalbe, K. *Řízení projektů v IT*. 1. Praha: Computer Press, Albatros Media a.s., 2016. ISBN 9788025147788.

[19] Singh, A. *What Is Rapid Application Development (RAD)?* 2019. Dostupné z: https://blog.capterra.com/what-is-rapid-application-development/.

Seznam obrázků

Seznam tabulek